

Oracle® Fusion Cloud EPM

REST API for Oracle Enterprise Performance Management Cloud



E96323-61



Oracle Fusion Cloud EPM REST API for Oracle Enterprise Performance Management Cloud,
E96323-61

Copyright © 2017, 2022, Oracle and/or its affiliates.

Primary Author: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Documentation Accessibility

Documentation Feedback

1 Implementation Best Practices for EPM Cloud REST APIs

2 About the REST APIs for EPM Cloud

About REST API for Oracle Enterprise Performance Management Cloud	2-1
EPM Cloud REST API Compatibility	2-2
About the Samples	2-13
Audience	2-14
Prerequisites	2-14
URL Structure	2-15

3 OAuth 2 and Basic Authentication for EPM Cloud REST APIs

Authentication with OAuth 2 - Only for OCI (Gen 2) Environments	3-1
Basic Authentication - for Classic and OCI (Gen 2) Environments	3-7

4 Sample Integration Scenarios

Scenario 1: Import Metadata into Applications	4-1
Scenario 2: Import Data, Run a Calculation Script, and Copy Data from a Block Storage Database to an Aggregate Storage Database	4-2
Scenario 3: Export and Download Metadata and Data	4-3
Scenario 4: Remove Unnecessary Files from a Service Instance	4-4
Scenario 5: Archive Backups from the Service to Onpremise	4-5
Scenario 6: Refreshing the Application	4-5
Scenario 7: Cloning an Instance	4-6
Scenario 8: Sample Starter Kit for Consultants - Business Intelligence Cloud Integration	4-7

5 Quick Reference Table – REST API Resource View

6 REST Resources and Methods

Supported REST Methods	6-1
REST API Methods	6-2
Error Handling	6-2
Versioning	6-2
Current REST API Version	6-3
Status Codes	6-3

7 Planning REST APIs

URL Structure for Planning	7-1
Resources and Available Actions	7-1
Getting API Versions for Planning	7-2
Get REST API Versions for Planning	7-2
Get Information about a Specific REST API Version for Planning	7-4
Manage Jobs	7-5
Get Job Definitions	7-6
Execute a Job	7-9
Rules	7-11
Ruleset	7-12
Plan Type Map	7-18
Import Data	7-19
Export Data	7-22
Import Metadata	7-25
Export Metadata	7-27
Cube Refresh	7-28
Clear Cube	7-29
Administration Mode	7-32
Compact Cube	7-33
Restructure Cube	7-34
Merge Data Slices	7-35
Optimize Aggregation	7-36
Import Security	7-39
Export Security	7-42
Export Audit	7-44

Export Job Console	7-47
Sort Members	7-54
Import Exchange Rates	7-57
Auto Predict	7-58
Import Cell-Level Security	7-60
Export Cell-Level Security	7-63
Import Valid Intersections	7-65
Export Valid Intersections	7-68
Execute a Report Bursting Definition	7-70
Execute Job Code Samples	7-72
Retrieve Job Status	7-74
Retrieve Job Status Details	7-76
Retrieve Child Job Status Details	7-78
Working with Members	7-81
Add Member	7-81
Get Member	7-83
Get Applications	7-85
Manage Planning Units	7-86
List All Planning Units	7-87
Get Planning Unit History and Annotations	7-91
Get a Planning Unit Owner Photo	7-94
Get Planning Unit Promotional Path	7-95
Get Available Planning Unit Actions	7-97
Get Filters with All Possible Values	7-99
Change Planning Unit Status	7-101
Get User Preferences	7-102
Working with Data Slices	7-103
Import Data Slices	7-104
Export Data Slices	7-106
Clear Data Slices	7-117
Getting and Setting Substitution Variables for Planning	7-119
Get All Substitution Variables Defined for the Application	7-120
Get a Substitution Variable Defined for the Application	7-122
Create or Update All Substitution Variables Defined for the Application	7-123
Get Substitution Variables Defined at the Plan Type Level	7-124
Get Derived Substitution Variables at the Plan Type Level	7-126
Get a Substitution Variable Defined at the Plan Type Level	7-127
Get a Derived Substitution Variable Defined at the Plan Type Level	7-128
Create and Update Substitution Variables at the Plan Type Level	7-130
Working with Connections	7-131
View a Connection	7-131

View All Connections	7-133
Update a Connection	7-135

8 Migration REST APIs

URL Structure for Migration	8-3
Getting API Versions for Migration APIs	8-3
Get REST API Versions for Migration	8-4
Get Information About a Specific REST API Version for Migration	8-7
Upload and Download Files	8-11
Upload	8-11
Download	8-17
View and Delete Files	8-28
List Files (v11.1.2.3.600)	8-28
List Files (v2)	8-31
Delete Files (v11.1.2.3.600)	8-33
Delete Files (v2)	8-36
Delete Files (v3)	8-38
Manage Services	8-40
Get Information About All Services	8-40
Run Recreate on a Service (11.1.2.3.600)	8-44
Run Recreate on a Service (v2)	8-50
Restart the Service Instance (v1)	8-67
Restart the Service Instance (v2)	8-71
Manage Application Snapshots	8-85
Get Information About All Application Snapshots	8-86
Get Information About a Specific Application Snapshot	8-87
Get Information about a Specific Application Snapshot Sample Code	8-90
Use the Application Snapshot Service	8-92
Upload Application Snapshot	8-93
Download Application Snapshot	8-97
Copy Application Snapshot (v1)	8-99
Copy Application Snapshot (v2)	8-104
Rename Application Snapshot (v1)	8-106
Rename Application Snapshot (v2)	8-108
List Backups - Only for OCI (Gen 2) Environments	8-110
Restore Backup - Only for OCI (Gen 2) Environments	8-111
View or Update the IP Allowlist - Only for OCI (Gen 2) Environments	8-114
View the IP Allowlist - Only for OCI (Gen 2) Environments	8-114
Update the IP Allowlist - Only for OCI (Gen 2) Environments	8-115
Copy to and from the Object Store	8-118

Copy to Object Store (v1)	8-118
Copy from Object Store (v1)	8-121
Copy to Object Store (v2)	8-124
Copy from Object Store (v2)	8-128
Provide Feedback (v11.1.2.3.600)	8-131
Provide Feedback (v2)	8-134
LCM Export (v1)	8-137
LCM Export (v2)	8-142
LCM Import (v1)	8-146
LCM Import (v2)	8-153
Manage Permission for Manual Access to Database (v1)	8-158
Manage Permission for Manual Access to Database (v2)	8-159
Set Encryption Key (v1)	8-162
Set Encryption Key (v2)	8-164
Migration Status Codes	8-166
Copy a File Between Instances (v1)	8-167
Copy a File Between Instances (v2)	8-169
Skip Updates (v1)	8-171
Skip Updates (v2)	8-173
Send Email (v1)	8-175
Send Email (v2)	8-178
Clone an Environment	8-181
Export Essbase Data (v2)	8-187

9 Viewing and Setting the Daily Maintenance Window Time

Getting and Setting the Daily Maintenance Window Time	9-1
Get the Build Version and Daily Maintenance Time (v1)	9-1
Get the Build Version and Daily Maintenance Window Time (v2)	9-5
Setting the Daily Maintenance Time (v1)	9-7
Setting the Daily Maintenance Time (v2)	9-10
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v1)	9-12
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)	9-16

10 Managing Users

Manage Users	10-1
Add Users to an Identity Domain	10-3
Remove Users from an Identity Domain	10-9
Assign Users to a Predefined Role or Application Role	10-13
Remove Users' Role Assignment	10-21

Update Users	10-29
Add Users To a Group	10-33
Remove Users from a Group	10-38
Add a User To a Batch of Groups	10-43
Remove a User from a Batch of Groups	10-48
Add Groups	10-53
Remove Groups	10-57
User Group Report	10-62
User Access Report (v1)	10-66
User Access Report (v2)	10-70
User Audit Report (v1)	10-73
User Audit Report (v2)	10-76
Role Assignment Report	10-79
Role Assignment Audit Report for OCI (Gen 2) Environments	10-83
Invalid Login Report for OCI (Gen 2) Environments	10-87
Group Assignment Audit Report	10-92
Adding Users to a Team for Account Reconciliation	10-95
Adding Users to a Team for Financial Consolidation and Close and Tax Reporting	10-98
Removing Users from a Team for Account Reconciliation	10-101
Removing Users from a Team for Financial Consolidation and Close and Tax Reporting	10-104

11 Reporting REST APIs

Generate Report for Account Reconciliation	11-1
Generate Report for Financial Consolidation and Close and Tax Reporting	11-5
Generate User Details Report for Account Reconciliation	11-9
Generate User Details Report for Financial Consolidation and Close and Tax Reporting	11-12
Retrieve Job Status for a Report	11-15
Execute Reports for Data Management	11-17

12 Data Management REST APIs

URL Structure for Data Management	12-1
Getting API Versions for Data Management APIs	12-1
Get API Versions for Data Management APIs	12-1
Get Information about a Specific API Version for Data Management APIs	12-3
Running Data Rules	12-4
Lock and Unlock POV	12-8
Integration Job Type	12-11
Import Data Mapping	12-22
Running Batch Rules	12-24

Export Data Mapping	12-26
Export Data Management	12-28
Import Data Management	12-30

13 Account Reconciliation APIs

URL Structure for Account Reconciliation	13-1
Getting API Versions for Account Reconciliation REST APIs	13-1
Get API Versions for Account Reconciliation REST APIs	13-1
Get Information about a Specific API Version for Account Reconciliation REST APIs	13-3
Execute a Job in Account Reconciliation	13-5
Change Period Status (Reconciliation Compliance)	13-6
Create Reconciliation (Reconciliation Compliance)	13-8
Import Pre-Mapped Balances (Reconciliation Compliance)	13-10
Import Pre-Mapped Transactions (Reconciliation Compliance)	13-11
Import Balances (Reconciliation Compliance)	13-13
Import Profiles (Reconciliation Compliance)	13-15
Import Rates (Reconciliation Compliance)	13-17
Import Pre-Mapped Transactions (Transaction Matching)	13-18
Monitor Reconciliations (Reconciliation Compliance)	13-20
Import Reconciliation Attributes (Reconciliation Compliance)	13-22
Run Auto Match (Transaction Matching)	13-25
Purge Transactions (Transaction Matching)	13-27
Retrieve Job Status for Account Reconciliation	13-30
Export Application Properties	13-31
Import Application Properties	13-34
Export Background Image	13-35
Import Background Image	13-37
Export Logo Image	13-39
Import Logo Image	13-41
Working with Connections in Account Reconciliation	13-42
Create a Connection	13-43
View All Connections	13-44
Update a Connection	13-46
Delete a Connection	13-47
Set Application Access Level	13-48
Retrieve Application Access Level	13-50

14 Financial Consolidation and Close REST APIs

Getting API Versions for Financial Consolidation and Close APIs	14-1
---	------

Getting API Versions for Financial Consolidation and Close	14-1
Get Information about a Specific API Version for Financial Consolidation and Close APIs	14-1
Import Supplemental Collection Data for Financial Consolidation and Close	14-3
Perform Journal Actions for Financial Consolidation and Close	14-6
Perform Journal Period Updates for Financial Consolidation and Close	14-8
Monitor Enterprise Journals for Financial Consolidation and Close	14-10
Retrieve Journals for Financial Consolidation and Close	14-12
Retrieve Journal Details for Financial Consolidation and Close	14-16
Copy Data	14-19
Clear Data	14-21
Deploy Form Templates	14-23
Deploy Task Manager Templates	14-30
Update Task Status for Event Monitoring	14-33
Execute an Enterprise Journals Job	14-34
Retrieve Enterprise Journals for Financial Consolidation and Close	14-41
Retrieve Enterprise Journal Content for Financial Consolidation and Close	14-43
Retrieve Enterprise Journal Content by Year and Period for Financial Consolidation and Close	14-45
Update Enterprise Journal Posting Status for Financial Consolidation and Close	14-48

15 Enterprise Profitability and Cost Management REST APIs

URL Structure for Enterprise Profitability and Cost Management	15-1
Getting API Versions for Enterprise Profitability and Cost Management	15-2
Getting Information About a Specific REST API Version for Enterprise Profitability and Cost Management	15-2
Calculate Model	15-3
Clear Data By Point Of View	15-7
Copy Data by Point of View	15-10
Delete Point of View	15-13
Validate Model	15-15

16 Profitability and Cost Management REST APIs

URL Structure for Profitability and Cost Management	16-1
Getting API Versions for Profitability and Cost Management REST APIs	16-1
Get API Versions for Profitability and Cost Management REST APIs	16-2
Java Sample – GetRestAPIVersionsInfo.java for Profitability and Cost Management	16-3
cURL Sample – GetRestAPIVersionInfo.sh for Profitability and Cost Management	16-3
Groovy Sample – GetRestAPIVersionsInfo.groovy for Profitability and Cost Management	16-4

Get Information about a Specific API Version for Profitability and Cost Management	16-5
Apply Data Grants	16-6
Java Sample – applyDataGrants.java for Profitability and Cost Management	16-7
cURL Sample – ApplyDataGrants.sh for Profitability and Cost Management	16-8
Groovy Sample – ApplyDataGrants.groovy for Profitability and Cost Management	16-8
Create File-Based Application	16-9
Java Sample – CreateFlatFileApplication.java for Profitability and Cost Management	16-10
cURL Sample – CreateFlatFileApplication.sh for Profitability and Cost Management	16-11
Groovy Sample – CreateFlatFileApplication.groovy for Profitability and Cost Management	16-11
Deploy ML Cube	16-12
Java Sample – DeployCube.java for Profitability and Cost Management	16-14
cURL Sample – DeployCube.sh for Profitability and Cost Management	16-14
Groovy Sample – DeployCube.groovy for Profitability and Cost Management	16-15
Copy ML POV Data	16-15
Java Sample – CopyPOV.java for Profitability and Cost Management	16-17
cURL Sample – CopyPOV.sh for Profitability and Cost Management	16-18
Java Sample – CopyPOV.java for Profitability and Cost Management	16-18
Enable File-Based Application	16-19
Java Sample – EnableApplication.java for Profitability and Cost Management	16-20
cURL Sample – EnableApplication.sh for Profitability and Cost Management	16-21
Groovy Sample – EnableApplication.groovy for Profitability and Cost Management	16-21
Essbase Data Load for Profitability and Cost Management	16-22
Java Sample – EssbaseDataLoad.java for Profitability and Cost Management	16-23
cURL Sample – EssbaseDataLoad.sh for Profitability and Cost Management	16-24
Groovy Sample – EssbaseDataLoad.groovy for Profitability and Cost Management	16-24
Export Template for Profitability and Cost Management	16-25
Java Sample – ExportTemplate.java for Profitability and Cost Management	16-26
cURL Sample – ExportTemplate.sh for Profitability and Cost Management	16-27
Groovy Sample – ExportTemplate.groovy for Profitability and Cost Management	16-27
Export Query Results	16-28
Java Sample – ExportQueryResult.java for Profitability and Cost Management	16-30
cURL Sample – ExportQueryResult.sh for Profitability and Cost Management	16-31
Groovy Sample – ExportQueryResult.groovy for Profitability and Cost Management	16-31
Generate Program Documentation Report	16-32
Java Sample – GeneratePrgrmDocReport.java for Profitability and Cost Management	16-33
cURL Sample – GeneratePrgDocReport.sh for Profitability and Cost Management	16-34
Groovy Sample – GeneratePrgrmDocReport.groovy for Profitability and Cost Management	16-35
Generate Program Documentation Report - Run as a Job	16-35
Java Sample – GeneratePrgrmDocReport.java for Profitability and Cost Management	16-37
cURL Sample – GeneratePrgDocReport.sh for Profitability and Cost Management	16-38

Groovy Sample – GeneratePrgrmDocReport.groovy for Profitability and Cost Management	16-38
Import Template for Profitability and Cost Management	16-39
Java Sample – ImportTemplate.java for Profitability and Cost Management	16-40
cURL Sample – ImportTemplate.sh for Profitability and Cost Management	16-41
Groovy Sample – ImportTemplate.groovy for Profitability and Cost Management	16-42
Merge Slices for Profitability and Cost Management	16-42
Optimize ASO Cube	16-44
Java Sample – OptimizeASOCube.java for Profitability and Cost Management	16-46
cURL Sample – OptimizeASOCube.sh for Profitability and Cost Management	16-46
Groovy Sample – OptimizeASOCube.groovy for Profitability and Cost Management	16-47
Retrieve Task Status for Profitability and Cost Management	16-47
Run ML Calculations	16-48
Java Sample – RunCalculation.java for Profitability and Cost Management	16-51
cURL Sample – RunCalculation.sh for Profitability and Cost Management	16-52
Groovy Sample – RunCalculation.groovy for Profitability and Cost Management	16-53
Run ML Clear POV	16-54
cURL Sample – ClearPOV.sh for Profitability and Cost Management	16-56
Groovy Sample – ClearPOV.groovy for Profitability and Cost Management	16-56
Java Sample – clearPOV.java for Profitability and Cost Management	16-57
Run ML Rule Balancing	16-57
Java Sample – RunRuleBalancing.java for Profitability and Cost Management	16-59
cURL Sample – RunRuleBalancing.sh for Profitability and Cost Management	16-60
Groovy Sample – RunRuleBalancing.groovy for Profitability and Cost Management	16-60
Update File-Based Application	16-61
Java Sample – UpdateDimension.java for Profitability and Cost Management	16-62
cURL Sample – UpdateDimension.sh for Profitability and Cost Management	16-63
Groovy Sample – UpdateDimension.groovy for Profitability and Cost Management	16-63
Update Dimensions As a Job	16-64
Java Sample – UpdateDimensionJob.java for Profitability and Cost Management	16-66
cURL Sample – UpdateDimensionJob.sh for Profitability and Cost Management	16-66
Groovy Sample – UpdateDimensionJob.groovy for Profitability and Cost Management	16-67

17 Narrative Reporting REST APIs

18 Enterprise Data Management Cloud REST APIs

A	Common Helper Functions for Java	
B	CSS Common Helper Functions for Java	
C	Common Helper Functions for cURL	
D	CSS Common Helper Functions for cURL	
E	CSS Common Helper Functions for Groovy	
F	REST API Examples with Postman	
	Example: Using REST APIs to Upload with Postman	F-1
	Example: Using REST APIs to Upload to an External Directory with Postman	F-3
	Example: Using REST APIs to Upload a Snapshot with Postman	F-4
G	Profitability and Cost Management Common Helper Functions	
	Profitability and Cost Management Common Helper Functions for Java	G-1
	Profitability and Cost Management Common Helper Functions for cURL	G-12
	Profitability and Cost Management Common Helper Functions for Groovy	G-16
H	Sample Starter Kit for Consultants - Integration with Business Intelligence Cloud Service	
	Installing the Scripting Engine and Deploying Demo Scripts	H-2
	SQL Application Express REST API client	H-2
	Business Intelligence REST API Client	H-5
	Planning REST API Client	H-7
	Helper Functions	H-8
	Integration of Planning to Business Intelligence Cloud Service	H-10
	Groovy Sample – PBCSBICSIntegration.groovy	H-11
	Groovy Sample – PbcRestClient.groovy	H-14
	Groovy Sample – PbcRestClient.groovy	H-25
	Groovy Sample – BicsRestClient.groovy	H-35
	Groovy Sample – ApexRestClient.groovy	H-48

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Documentation Feedback

To provide feedback on this documentation, click the feedback button at the bottom of the page in any Oracle Help Center topic. You can also send email to epmdoc_ww@oracle.com.

1

Implementation Best Practices for EPM Cloud REST APIs

Use the implementation best practices listed in this topic when working with the EPM Cloud REST APIs.

Best practices:

- Before using the REST APIs, complete the [prerequisites](#).
- Use the correct authentication, as described in [OAuth 2 and Basic Authentication for EPM Cloud REST APIs](#).
- Understand the [URL structure](#).
- Know how to [get the current REST API version](#).
- Review the [sample scenarios](#) to get started quickly.
- Be aware of [REST API compatibility](#).
- Use the [Quick Reference](#) to find all of the Oracle Enterprise Performance Management Cloud REST APIs at a glance.

2

About the REST APIs for EPM Cloud

Review these topics to learn about the REST APIs for Oracle Enterprise Performance Management Cloud and understand important prerequisites and authentication.

Overview of the REST APIs:

- [About REST API for Oracle Enterprise Performance Management Cloud](#)
- [Implementation Best Practices for EPM Cloud REST APIs](#)
- [EPM Cloud REST API Compatibility](#)
- [About the Samples](#)
- [Audience](#)
- [Prerequisites](#)
- [OAuth 2 and Basic Authentication for EPM Cloud REST APIs](#)

About *REST API for Oracle Enterprise Performance Management Cloud*

This guide describes REST APIs for Oracle Enterprise Performance Management Cloud.

These REST APIs allow service administrators and infrastructure consultants to perform administration tasks in EPM Cloud. This guide assumes that the audience has technical and functional expertise in using and working with REST APIs. See [Audience](#).

EPM Cloud includes these cloud services:

- Planning
- FreeForm
- Planning Modules
- Account Reconciliation
- Financial Consolidation and Close
- Enterprise Profitability and Cost Management
- Profitability and Cost Management
- Tax Reporting
- Narrative Reporting
- Oracle Enterprise Data Management Cloud

You can integrate EPM Cloud environments using:

- A set of REST APIs
- The EPM Automate Utility, a command line tool that is implemented on top of the REST APIs

- Groovy business rules, as described in [Oracle Enterprise Performance Management Cloud Groovy Rules Java API Reference](#).

EPM Cloud REST API Compatibility

This table summarizes the compatibility for Oracle Enterprise Performance Management Cloud REST APIs.

NOTE: These abbreviations are used in the column headings of this table: **PLN** (Planning), **FF** (FreeForm), **SWP** (Strategic Workforce Planning), **SP** (Sales Planning), **FCC** (Financial Consolidation and Close), **AR** (Account Reconciliation), **EPCM** (Enterprise Profitability and Cost Management), **PCM** (Profitability and Cost Management), and **TR** (Tax Reporting)

For detailed information about REST APIs, see [Quick Reference Table – REST API Resource View](#).

Table 2-1 REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
EPM Platform						
View a Connection	Yes	Yes		Yes		Yes
View all Connections	Yes	Yes		Yes		Yes
Update a Connection	Yes	Yes		Yes		Yes
Execute a Report Bursting Definition	Yes	Yes		Yes		Yes
Planning, FreeForm, Strategic Workforce Planning, and Sales Planning						
Getting API Versions for Planning	Yes	Yes		Yes		Yes
Get Information about a Specific REST API Version for Planning	Yes	Yes		Yes		Yes
Get Job Definitions	Yes	Yes		Yes		Yes
Execute a Job	Yes	Yes		Yes		Yes
Retrieve Job Status	Yes	Yes		Yes		Yes
Retrieve Job Status Details	Yes	Yes		Yes		Yes
Retrieve Child Job Status Details	Yes	Yes		Yes		Yes
Add Member	Yes	Yes		Yes		Yes
Get Member	Yes	Yes		Yes		Yes

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Get Applications	Yes	Yes		Yes		Yes
List All Planning Units	Yes	Yes				
Get Planning Unit History and Annotations	Yes	Yes				
Get a Planning Unit Owner Photo	Yes	Yes				
Get Planning Unit Promotional Path	Yes	Yes				
Get Available Planning Unit Actions	Yes	Yes				
Get Filters with All Possible Values	Yes	Yes		Yes		
Change Planning Unit Status	Yes	Yes				
Get User Preferences	Yes	Yes		Yes		Yes
Import Data Slice	Yes	Yes		Yes		Yes
Export Data Slice	Yes	Yes		Yes		Yes
Clear Data Slice	Yes	Yes		Yes		Yes
Get All Substitution Variables Defined for the Application	Yes	Yes		Yes		Yes
Get a Substitution Variable Defined for the Application	Yes	Yes		Yes		
Create or Update All Substitution Variables Defined for the Application	Yes	Yes		Yes		
Get Substitution Variables Defined at the Plan Type Level	Yes	Yes		Yes		
Get Derived Substitution Variables at the Plan Type Level	Yes	Yes		Yes		
Get a Substitution Variable Defined at the Plan Type Level	Yes	Yes		Yes		
Get a Derived Substitution Variable Defined at the Plan Type Level	Yes	Yes		Yes		

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Create and Update Substitution Variables at the Plan Type Level	Yes	Yes		Yes		
Migration						
Get REST API Versions for Migration	Yes	Yes	Yes	Yes	Yes	Yes
Get Information about a Specific Version of Migration Sample Code	Yes	Yes	Yes	Yes	Yes	Yes
Upload	Yes	Yes	Yes	Yes	Yes	Yes
Download	Yes	Yes	Yes	Yes	Yes	Yes
List Files (v11.1.2.3.600)	Yes	Yes	Yes	Yes	Yes	Yes
List Files (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Delete Files (v11.1.2.3.600)	Yes	Yes	Yes	Yes	Yes	Yes
Delete Files (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Delete Files (v3)	Yes	Yes	Yes	Yes	Yes	Yes
Get Information About All Services	Yes	Yes	Yes	Yes	Yes	Yes
Run Recreate on a Service (11.1.2.3.600)	Yes	Yes	Yes	Yes	Yes	Yes
Run Recreate on a Service (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Restart the Service Instance (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Restart the Service Instance (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Get Information About All Application Snapshots	Yes	Yes	Yes	Yes	Yes	Yes
Get Information about a Specific Application Snapshot Sample Code	Yes	Yes	Yes	Yes	Yes	Yes
Upload Application Snapshot	Yes	Yes	Yes	Yes	Yes	Yes

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Download Application Snapshot	Yes	Yes	Yes	Yes	Yes	Yes
Copy Application Snapshot (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Rename Application Snapshot (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Rename Application Snapshot (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Copy Application Snapshot (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Copy to Object Store (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Copy to Object Store (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Copy from Object Store (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Copy from Object Store (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Provide Feedback (v11.1.2.3.600)	Yes	Yes	Yes	Yes	Yes	Yes
Provide Feedback (v2)	Yes	Yes	Yes	Yes	Yes	Yes
LCM Export (v1)	Yes	Yes	Yes	Yes	Yes	Yes
LCM Export (v2)	Yes	Yes	Yes	Yes	Yes	Yes
LCM Import (v1)	Yes	Yes	Yes	Yes	Yes	Yes
LCM Import (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Manage Permission for Manual Access to Database (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Migration Status Codes	Yes	Yes	Yes	Yes	Yes	Yes
Copy a File Between Instances (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Copy a File Between Instances (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Skip Updates (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Skip Updates (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Send Email (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Send Email (v2)	Yes	Yes	Yes	Yes	Yes	Yes

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Get the Build Version and Daily Maintenance Time (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Get the Build Version and Daily Maintenance Window Time (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Setting the Daily Maintenance Time (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Setting the Daily Maintenance Time (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Clone an Environment	Yes	Yes	Yes	Yes	Yes	Yes
Add Users to an Identity Domain	Yes	Yes	Yes	Yes	Yes	Yes
Remove Users from an Identity Domain	Yes	Yes	Yes	Yes	Yes	Yes
Assign Users to Predefined or Application Roles	Yes	Yes	Yes	Yes	Yes	Yes
Remove Users' Role Assignment	Yes	Yes	Yes	Yes	Yes	Yes
Add Users to an Identity Domain	Yes	Yes	Yes	Yes	Yes	Yes
Remove Users from an Identity Domain	Yes	Yes	Yes	Yes	Yes	Yes
Update Users	Yes	Yes	Yes	Yes	Yes	Yes
User Group Report	Yes	Yes	Yes	Yes	Yes	Yes
User Audit Report (v1)	Yes	Yes	Yes	Yes	Yes	Yes
User Audit Report (v2)	Yes	Yes	Yes	Yes	Yes	Yes
User Access Report (v1)	Yes	Yes	Yes	Yes	Yes	Yes

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
User Access Report (v2)	Yes	Yes	Yes	Yes	Yes	Yes
View the IP Allowlist - Only for OCI (Gen 2) Environments	Yes	Yes	Yes	Yes	Yes	Yes
Update the IP Allowlist - Only for OCI (Gen 2) Environments	Yes	Yes	Yes	Yes	Yes	Yes
Group Assignment Audit Report	Yes	Yes	Yes	Yes	Yes	Yes
Set Encryption Key (v1)	Yes	Yes	Yes	Yes	Yes	Yes
Set Encryption Key (v2)	Yes	Yes	Yes	Yes	Yes	Yes
Role Assignment Report	Yes	Yes	Yes	Yes	Yes	Yes
Add a User To a Batch of Groups	Yes	Yes	Yes	Yes	Yes	Yes
Remove a User from a Batch of Groups	Yes	Yes	Yes	Yes	Yes	Yes
Adding Users to a Team for Account Reconciliation			Yes			
Removing Users from a Team for Account Reconciliation			Yes			
Adding Users to a Team for Financial Consolidation and Close and Tax Reporting		Yes				Yes
Removing Users from a Team for Financial Consolidation and Close and Tax Reporting		Yes				Yes
List Backups - Only for OCI (Gen 2) Environments	Yes	Yes	Yes	Yes	Yes	Yes
Restore Backup - Only for OCI (Gen 2) Environments	Yes	Yes	Yes	Yes	Yes	Yes
Export Essbase Data (v2)	Yes	Yes		Yes		Yes
Data Management						

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Get API Versions for Data Management APIs	Yes	Yes	Yes	Yes	Yes	Yes
Get Information about a Specific API Version for Data Management APIs	Yes	Yes	Yes	Yes	Yes	Yes
Running Data Rules	Yes	Yes	Yes	Yes	Yes	Yes
Integration Job Type	Yes	Yes	Yes	Yes	Yes	Yes
Running Batch Rules	Yes	Yes	Yes	Yes	Yes	Yes
Import Data Mapping	Yes	Yes	Yes	Yes	Yes	Yes
Export Data Mapping	Yes	Yes	Yes	Yes	Yes	Yes
Export Data Management	Yes	Yes		Yes	Yes	Yes
Import Data Management	Yes	Yes		Yes	Yes	Yes
Lock and Unlock POV	Yes	Yes	Yes	Yes	Yes	Yes
Execute Reports for Data Management	Yes	Yes	Yes	Yes	Yes	Yes
Account Reconciliation						
Get API Versions for Account Reconciliation REST APIs			Yes			
Get Information about a Specific API Version for Account Reconciliation REST APIs			Yes			
Generate Report for Account Reconciliation			Yes			
Retrieve Job Status for a Report			Yes			
Generate User Details Report for Financial Consolidation and Close and Tax Reporting			Yes			

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Export Application Properties			Yes			
Import Application Properties			Yes			
Export Background Image			Yes			
Import Background Image			Yes			
Export Logo Image			Yes			
Import Logo Image			Yes			
Create a Connection			Yes			
View All Connections			Yes			
Update a Connection			Yes			
Delete a Connection			Yes			
Set Application Access Level			Yes			
Retrieve Application Access Level			Yes			
Reconciliation Compliance						
Execute a Job in Account Reconciliation			Yes			
Create Reconciliation (Reconciliation Compliance)			Yes			
Change Period Status (Reconciliation Compliance)			Yes			
Import Pre-Mapped Transactions (Reconciliation Compliance)			Yes			
Import Profiles (Reconciliation Compliance)			Yes			

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Import Rates (Reconciliation Compliance)			Yes			
Import Balances (Reconciliation Compliance)			Yes			
Import Reconciliation Attributes (Reconciliation Compliance)			Yes			
Monitor Reconciliations (Reconciliation Compliance)			Yes			
Generate Report for Account Reconciliation			Yes			
Transaction Matching						
Import Pre-Mapped Transactions (Transaction Matching)			Yes			
Run Auto Match (Transaction Matching)			Yes			
Retrieve Job Status for Account Reconciliation			Yes			
Generate Report for Account Reconciliation			Yes			
Financial Consolidation and Close						
Getting API Versions for Financial Consolidation and Close		Yes				
Get Information about a Specific API Version for Financial Consolidation and Close APIs		Yes				
Deploy Task Manager Templates		Yes				Yes

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Import Supplemental Collection Data for Financial Consolidation and Close		Yes				
Perform Journal Actions for Financial Consolidation and Close		Yes				
Perform Journal Period Updates for Financial Consolidation and Close		Yes				
Monitor Enterprise Journals for Financial Consolidation and Close		Yes				
Retrieve Journals for Financial Consolidation and Close		Yes				
Retrieve Journal Details for Financial Consolidation and Close		Yes				
Copy Data		Yes				
Clear Data		Yes				
Deploy Form Templates		Yes				
Deploy Task Manager Templates		Yes				Yes
Update Task Status for Event Monitoring		Yes				Yes
Execute an Enterprise Journals Job		Yes				
Retrieve Enterprise Journals for Financial Consolidation and Close		Yes				

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Retrieve Enterprise Journal Content for Financial Consolidation and Close		Yes				
Retrieve Enterprise Journal Content by Year and Period for Financial Consolidation and Close		Yes				
Update Enterprise Journal Posting Status for Financial Consolidation and Close		Yes				
Generate Report for Financial Consolidation and Close and Tax Reporting		Yes				Yes
Generate User Details Report for Financial Consolidation and Close and Tax Reporting		Yes				Yes
Retrieve Job Status for a Report		Yes				Yes
Enterprise Profitability and Cost Management						
Calculate Model				Yes		
Clear Data By Point Of View				Yes		
Copy Data by Point of View				Yes		
Delete Point of View				Yes		
Validate Model				Yes		
Profitability and Cost Management						
Apply Data Grants					Yes	
Copy ML POV Data					Yes	

Table 2-1 (Cont.) REST API Compatibility

Business Process and Task	PLN, FF, SWP, SP	FCC	AR	EPCM	PCM	TR
Create File-Based Application					Yes	
Deploy ML Cube					Yes	
Essbase Data Load for Profitability and Cost Management					Yes	
Generate Program Documentation Report					Yes	
Generate Program Documentation Report - Run as a Job					Yes	
Get Information about a Specific API Version for Profitability and Cost Management					Yes	
Get API Versions for Profitability and Cost Management REST APIs					Yes	
Merge Slices for Profitability and Cost Management					Yes	
Optimize ASO Cube					Yes	
Retrieve Task Status for Profitability and Cost Management					Yes	
Run ML Calculations					Yes	
Run ML Clear POV					Yes	
Run ML Rule Balancing					Yes	
Update Dimensions As a Job					Yes	
Update File-Based Application					Yes	

About the Samples

Samples are described for selected integration scenarios and REST API reference sections. A working knowledge of Java, cURL, and Groovy is required to understand these samples.

The Java samples in this guide are coded using pure Java instead of third-party libraries such as Apache HTTP Client. The only JAR files you will need outside of JDK will be for JSON parsing.

This document does not teach REST concepts. As a prerequisite, a prior knowledge of REST programming is required to understand the examples, samples, scenarios, and reference sections.

Audience

This guide is intended primarily as a tool for infrastructure consultants and administrators of EPM Cloud.

Infrastructure consultants can use the documentation to build custom integration to provide basic and innovative services on top of these cloud services, including Planning, Planning Modules, Account Reconciliation, and Profitability and Cost Management.

Service administrators use this documentation to perform selected administrative tasks using REST APIs and EPM Automate. Completing administrative tasks using EPM Automate and REST APIs as an alternative to using the user interface requires considerable technical and functional expertise. Only technically competent administrators should use this guide to perform these administrative tasks.

Prerequisites

Prerequisites to using the REST APIs and the EPM Automate Utility include the following:

- Access as a valid user to the cloud service with prerequisites set up for the service. See [Getting Started with Oracle Enterprise Performance Management as an Administrator](#) and [Getting Started with Oracle Enterprise Performance Management as a User](#).
- Technical and functional knowledge to understand and execute the EPM Automate Utility and REST APIs, and to administer the product.
- Knowledge of Java, cURL, Groovy, and REST programming.
- Jobs are required for many EPM Automate utility commands and REST APIs. Jobs are actions, such as importing or exporting data that can be started immediately or scheduled for a later time; for example, importing or exporting data. Be sure that you understand how to use jobs as described in [Managing Jobs](#).
- Data load rules define how Data Management loads data from a file. You must have predefined data load rules to load data using REST APIs and EPM Automate Utility. You can also load data using batches defined in Data Management. Using a batch, Service Administrators can combine many load rules in a batch and execute them in serial or parallel mode.
- Business rules are required for some jobs. For example for Planning, you use Calculation Manager to create business rules, which are then deployed into a Planning application. Learn about business rules in [Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud](#).

URL Structure

For the URL structure to use, see the topic for the REST API:

- [Planning and Budgeting REST API URL](#)
- [Migration REST API URL](#)
- [Data Management REST API URL](#)
- [Account Reconciliation REST API URL](#)
- [Enterprise Profitability and Cost Management REST API URL](#)
- [Profitability and Cost Management REST API URL](#)

The URL examples shown in this guide use the Classic format.

The Classic URL structure begins with this path:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
```

where *SERVICE_NAME* is the name of the service, *TENANT_NAME* is the identity domain, *SERVICE_TYPE* is *epm* (for environments created many years ago, it could be *pbcs*), and *dcX* is the data center (for example, *us2*).

For OCI (Gen 2), URLs are in a slightly different format, with *.ocs.* before *oraclecloud.com*, for example:

```
https://<SERVICE_NAME>-<TENANT_NAME>.epm.<dcX>.ocs.oraclecloud.com/
```

where *dcX* is the data center region (for example, *us-phoenix-1*).

Note: Oracle does not authorize or support the use of REST APIs with the path token `"/internal/"` in the URL.

For details on the URL structure, see [Differences Between Classic and OCI EPM Cloud Environments](#). To learn about accessing Oracle Cloud and Oracle Enterprise Performance Management Cloud, see [Getting Started with Oracle Cloud](#) and [Getting Started with Oracle Enterprise Management Cloud for Administrators](#).

3

OAuth 2 and Basic Authentication for EPM Cloud REST APIs

All HTTP requests to the REST APIs require authentication. Review these topics to understand OAuth 2 and basic authentication for EPM Cloud REST APIs.

The REST APIs for EPM Cloud support authentication with OAuth 2 and basic authentication:

- [Authentication with OAuth 2 - Only for OCI \(Gen 2\) Environments](#)
- [Basic Authentication - for Classic and OCI \(Gen 2\) Environments](#)

Authentication with OAuth 2 - Only for OCI (Gen 2) Environments

In EPM Cloud environments on Oracle Cloud Infrastructure (OCI) / Gen 2 architecture, you can use an OAuth 2 access token to issue REST APIs on EPM Cloud to satisfy the requirement of avoiding the use of passwords in your environment.

Setting Up Authentication with OAuth 2

In order to access EPM Cloud REST APIs with OAuth 2, an EPM Cloud Service Administrator has to request the Domain Administrator to set up an OAuth 2 client and provide the Identity Domain Cloud Service (IDCS) URL, application scope, and Client ID.

The Domain Administrator registers the REST API client following the procedure described below.

Register the Client

The first step is to update the service provider configuration to authorize requests from the REST client application. As a security measure, any client application that accesses Oracle Cloud resources must be authorized to do so. A domain administrator provides this authorization by registering a client and providing the appropriate registration information to the client's users.

Clients can be public (outside a data center) or confidential. Public clients are assigned a client_id. Confidential clients also have a client secret in addition to a client_id. Clients are also authorized to access particular scopes. The type of application you select determines the allowed grant types available to request access tokens.

Client applications require an access token in order to access server resources. To obtain an access token, the client implements one of the IDCS supported access grant types [as listed here](#). The choice of access grant type to implement is based on the requirements and desired functionality of the REST API client.

The next section highlights the steps to create a sample public OAuth 2 client that accesses the EPM Cloud Get Daily Maintenance Window REST API.

For details, see the Oracle Identity Cloud Service documentation:

- [Supported Access Grant Types](#)
- [Scopes](#)

The Identity Cloud Service domain administrator follows the steps in this topic to create a public client in the Identity Cloud Service Administrator console for the Application Scope for the requested client. The domain administrator then shares the Identity Cloud Service application URL, application scope, and client ID with the EPM Cloud Service Administrator.

Identity Cloud Service Administrator tasks to register a REST client:

1. Log in to the Identity Cloud Service Administrator console as a Domain Administrator.
 2. Configure the token properties for protected OAuth 2 REST APIs:
 - a. From the **Applications** drawer, click **Oracle Cloud Services**, and then choose the Cloud Service.
 - b. On the **Configuration** tab, under **Resources**, select the checkbox for **Is Refresh Token Allowed**.
 - c. **Optional:** Change the **Access Token Expiration** and **Refresh Token Expiration**. Oracle recommends 3600 (1 hour) as the value of Access Token Expiration and 604,800 (7 days) as the value of Refresh Token Expiration.
 - d. Click **Save** on the upper-right to save changes.
 3. From the **Applications** drawer, click **Add** at the top of the page.
 4. In the **Add Application** dialog box, select **Mobile Application**.
 5. In the **App Details** section, enter a name for the REST client.
 6. **Optional:** Add other details.
 7. Click **Next**.
 8. In the **Authorization** section, under **Allowed Grant Types**, select the checkboxes for **Refresh Token** and **Device Code**.
 9. Under **Token Issuance Policy** section, click **Add Scope**.
 - a. On the **Select Scope** dialog box, choose the **Resource** (EPM Cloud environment) that this client will access. Note the **Scope** for this Resource.
 - b. Click **Add** to close the pop-up.
 - c. At the top of the page, click **Next**.
 - d. Click **Finish**.
- The **Application Added** dialog box shows the generated client ID. This information also appears on the **Configuration** tab in the **Details** section for the application. Click **Close**.
10. Click **Activate** to activate the client, and then click **OK**.

For details, see the Oracle Identity Cloud Service documentation:

- [Adding Applications](#)
- [Add a Mobile Application](#)

The Domain Administrator provides the IDCS URL, scope (obtained in Step 9.a.) and client ID (obtained in Step 9.d.) to the EPM Cloud Service Administrator.

After the Domain Administrator has registered the REST client and provided the IDCS URL, scope, and client ID, the EPM Cloud Administrator executes the following steps to get a valid refresh token.

EPM Cloud Service Administrator tasks to get a refresh token:

1. Issue the following unauthenticated request to the Identity Cloud Service URL:

```
curl --location --request POST 'https://tenant-base-url/oauth2/v1/device' \
  \
  --header 'Content-Type: application/x-www-form-urlencoded;charset=utf-8' \
  --data-urlencode 'response_type=device_code' \
  --data-urlencode
  'scope=urn:opc:serviceInstanceID=507199212urn:opc:resource:consumer::all
  offline_access' \
  --data-urlencode 'client_id=75ce2d27564d4432863927946c6bafc7'
```

Here, the value of `tenant-base-url` is the IDCS URL provided by the Domain Administrator. Similarly, the values for `scope` and `client_id` are also provided by the Domain Administrator.

The response contains a device code, user code, and verification URI:

```
{
  "expires_in": 300,
  "device_code": "4d03f7bc-f7a5-4795-819a-5748c4801d35",
  "user_code": "SDFGHJKL",
  "verification_uri": "https://tenant-base-url/ui/v1/device"
}
```

Note: Steps 2 and 3 are time-sensitive because the user code and device code expire 300 seconds (5 minutes) after creation. If the codes expire before these steps can be completed, restart from Step 1.

2. Open the `verification_uri` in a web browser.
 - a. If prompted for credentials, enter the credentials. This can be the credentials stored in IDCS or in a SAML 2.0-based Identity Provider.
 - b. When prompted for code, enter the `user_code` from the response payload.
 - c. When the Successful message is displayed, close the browser window or tab.
3. Issue the following request to the Identity Cloud Service URL:

```
curl --location --request POST 'https://tenant-base-url/oauth2/v1/token' \
  --header 'Content-Type: application/x-www-form-urlencoded;charset=utf-8' \
  --data-urlencode 'grant_type=urn:ietf:params:oauth:grant-
  type=device_code' \
  --data-urlencode 'device_code=4d03f7bc-f7a5-4795-819a-5748c4801d35' \
  --data-urlencode 'client_id=75ce2d27564d4432863927946c6bafc7'
```

Here the value of `tenant-base-url` is the IDCS URL provided by the Domain Administrator, the `device_code` value is obtained in Step 1, and `client_id` is provided by the Domain Administrator.

The response contains a refresh token:

```
{
  "access_token": "eyJ4NXQjUzI.....evRJChXTRfzn6WlCw",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "AQIDBAWF1....RVkxNCB7djF9NCA="
}
```

Note the `client_id` and `refresh_token` because they are required in the next step by the REST client.

Use of OAuth 2 in the REST Client

The REST client uses the refresh token and client id to get access token and uses it for accessing REST APIs. It also makes sure that the refresh token, access token, and client ID are stored securely.

Obtain a Valid Access Token

The REST client should use the refresh token grant type to get a new access token and a new refresh token.

To obtain a valid access token, the REST client issues the following request to the IDCS URL:

```
curl --location --request POST 'https://tenant-base-url/oauth2/v1/
token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
--data-urlencode 'client_id=75ce2d27564d4432863927946c6bafc7' \
--data-urlencode 'refresh_token=AQIDBAWF1....FVkxNCB7djF9NCA='
```

Here the value of `tenant-base-url` is the IDCS URL provided by the Domain Administrator, the `refresh_token` value is obtained from the secure store where it was stored in the previous step, and `client_id` is provided by the Domain Administrator.

Example response:

```
{
  "access_token": "aQj5M4QjUkI.....abSjZaa86PlseS4lrt7R2",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "AAyyilyBAWD4....FVkxefd8kjoJr6HJPA="
}
```

The REST client saves the refresh token for future use (see [Secure and Protect the Token and Client ID](#)) and uses the access token (see [Use the Access Token](#)).

Use the Access Token

In order to access an application (environment), the REST client must provide the access token (obtained in the previous step) in the authorization header as follows:

```
Authorization: Bearer <access token>
```

For example, to get the Automated Maintenance Window start time, the client application submits a GET request to this endpoint: `/interop/rest/v1/services/dailymaintenance` using the access token in the authorization header.

```
curl --location --request GET 'https://tenant-base-url/interop/rest/v1/
services/dailymaintenance' \
--header "Authorization: Bearer "aQj5M4QjUkI.....abSjZaa86PlseS4lrt7R2"
```

Secure and Protect the Tokens and Client_ID

An access token is used instead of user credentials to access the resources on EPM Cloud. A refresh token and client ID are used to get a new access token and a new refresh token. Thus, to ensure security of EPM Cloud, it is important to securely encrypt and store the `client_id` and all tokens. The REST client must securely store the access token, refresh token, and `client_id`.

Frequently Asked Questions

Can the expiry time of a refresh token be modified?

The expiry time of the refresh token is configurable in Identity Cloud Service by the Domain Administrator on a per EPM Cloud environment basis. The client requesting the refresh token cannot modify the expiry time or duration. The client should request a new refresh token before the old one expires, in order to not repeat the initial setup again. The default expiry period is 604800 seconds, which is 7 days.

What error is returned when the refresh token has expired?

Executing a refresh token grant flow with an expired refresh token results in a *400 Bad Request* response and the following payload:

```
{ "error": "invalid_grant",
  "error_description": "Token is expired for client :
75ce2d27564d4432863927946c6bafc7",
  "ecid": "UsbMB0KCV00000000"
}
```

What error is returned when the refresh token is invalid?

Executing a refresh token grant flow with an invalid refresh token results in a *400 Bad Request* response and the following payload:

```
{ "error": "invalid_grant",
  "error_description": "The given token in the request is invalid",
  "ecid": "UsbMB0KCV00000000"
}
```

What errors are returned for other issues?

A *400 Bad Request* response with the following payloads are returned when there is an error:

Invalid request (for example, not all request parameters are supplied):

```
{
  "error": "invalid_request",
  "error_description":
"The request contains invalid parameters or values"
}
```

Invalid grant (for example, using a token that has already been used) :

```
{
  "error": "invalid_grant",
  "error_description": "The token has already been consumed"
}
```

Invalid scope (for example, providing invalid scope):

```
{
  "error": "invalid_scope",
  "error_description": "Invalid scope"
}
```

What error is returned when an invalid or expired access token is provided?

When an invalid or expired access token is provided in a request, the server responds with a *401 Unauthorized* response with the following HTML in the payload:

```
<html>
  <head>,
    <title>401 Authorization Required</title>
  </head>
  <body>
    <center>
      <h1>401 Authorization Required</h1>
    </center>
    <hr>
    <center>nginx/1.16.1</center>
  </body>
</html>
```

Can a token be requested with multiple scopes?

Multiple scopes across different resources (EPM Cloud environments) are not supported by Identity Cloud Service. Each token request can support only one resource. Requests for multiple scopes within the same resource are supported with

space delimited scopes. Requesting multiple scopes across different resources results in a *400 Bad Request* response with the following payload:

```
{
  "error": "invalid_scope",
  "error_description": "Invalid scope"
}
```

What information is logged in the access log with OAuth?

The access log shows the user name, just as it does with basic authorization. The client ID and access token are not logged.

Basic Authentication - for Classic and OCI (Gen 2) Environments

In EPM Cloud environments on Classic or OCI (Gen 2) architecture, the REST APIs support basic authentication (name and password).

If your environment is on Classic Oracle Cloud Infrastructure, use a username in the format `identitydomain.username`.

If your environment is on OCI (Gen 2) Oracle Cloud Infrastructure, you can use basic authentication by supplying only `username` (without identity domain).

HTTP requests to Oracle Enterprise Performance Management Cloud should supply HTTP Basic Authentication credentials through the `Authorization` header.

Finding Your Identity Domain

If your environment is on Classic Oracle Cloud Infrastructure, ensure that you are correctly specifying your identity domain when logging into an environment using REST APIs.

Use one of these methods to identify your identity domain:

- Look in the Activity Report for your environment. The name of the identity domain is displayed at the top left corner of the Activity Report. See "About the Activity Report" in *Getting Started with EPM Cloud for Administrators*.

- Derive the identity domain name from the URL that you use to access the environment. For example, in this fictitious URL, `https://epm-exampleDomain.epm.dataCenter.oraclecloud.com/`, the identity domain name is **exampleDomain**.

The test and production environments of a subscription usually share the same identity domain. For example, for test environments like as this fictitious URL, `https://epm-test-exampleDomain.epm.dataCenter.oraclecloud.com/`, the identity domain can be derived as the string enclosed between "test-" and the next ".", in this case, **exampleDomain**.

For production environments like this fictitious URL, `https://epm-exampleDomain.epm.dataCenter.oraclecloud.com/`, the identity domain can be derived as the string enclosed between the first "-" and next ".", in this case, **exampleDomain**.

Note: In this guide, URLs are shown in the following format: `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/`

When using PowerShell to call EPMCloud REST API, ensure that the Authorization header is always specified. For example:

```
$headers = @{  
"Authorization" = "Basic " +  
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(  
$userName+": "+$userPassword))  
}  
Invoke-RestMethod -Method 'Get' -Uri $url -Headers $headers
```


4

Sample Integration Scenarios

This section provides selected sample scenarios for EPM Cloud REST APIs to help you get started.

Related Topics

- [Scenario 1: Import Metadata into Applications](#)
This scenario shows how to use EPM Cloud REST APIs to import metadata into applications.
- [Scenario 2: Import Data, Run a Calculation Script, and Copy Data from a Block Storage Database to an Aggregate Storage Database](#)
This scenario shows how to use the EPM Cloud REST APIs to import data, run a calculation script, and copy data from a block storage database to an aggregate storage database.
- [Scenario 3: Export and Download Metadata and Data](#)
This scenario shows how to use the EPM Cloud REST APIs to export and download metadata and data.
- [Scenario 4: Remove Unnecessary Files from a Service Instance](#)
This scenario shows how to use the EPM Cloud REST APIs to remove unnecessary files from a service instance.
- [Scenario 5: Archive Backups from the Service to Onpremise](#)
This scenario shows how to use the EPM Cloud REST APIs to archive backups from the service to onpremise.
- [Scenario 6: Refreshing the Application](#)
This scenario shows how to use the EPM Cloud REST APIs to refresh the application.
- [Scenario 7: Cloning an Instance](#)
This scenario shows how to use the EPM Cloud REST APIs to clone an instance.
- [Scenario 8: Sample Starter Kit for Consultants - Business Intelligence Cloud Integration](#)
This scenario provides a sample starter kit for consultants to integrate with Oracle Business Intelligence Cloud.
- [Scenario 9: Using Groovy Business Rules to Call REST APIs from Oracle and Other Companies](#)
The scenario shows how to use the Oracle Enterprise Performance Management Cloud Groovy object model to call Oracle REST APIs and REST APIs developed by other companies.

Scenario 1: Import Metadata into Applications

This scenario shows how to use EPM Cloud REST APIs to import metadata into applications.

Example 4-1 Java

```
public void integrationScenarioImportMetadataIntoApplication() throws  
Exception {  
    uploadFile("accounts.zip");  
}
```

```
        executeJob("IMPORT_METADATA", "accountMetadata",
"{importZipFileName:accounts.zip}");
        executeJob("CUBE_REFRESH", null, null);
    }
```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: see [Java Sample – UploadFile.java](#) and [Java Sample – ExecuteJob.java](#) in [Upload and Download Files](#).

Example 4-2 cURL

```
funcIntegrationScenarioImportMetadataIntoApplication() {
    funcUploadFile "DemoApplication_HSS_Vision.zip"
    funcExecuteJob "IMPORT_METADATA" "accountMetadata"
"{importZipFileName=accounts.zip}"
    funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}
```

Example 4-3 Groovy

```
def integrationScenarioImportMetadataIntoApplication() {
    uploadFile("DemoApplication_HSS_Vision.zip")
    executeJob("IMPORT_METADATA", "accountMetadata",
"importZipFileName=accounts.zip");
    executeJob("CUBE_REFRESH", "cubeRefresh", null);
}
```

Common functions: See [CSS Common Helper Functions for Groovy](#)

Scenario 2: Import Data, Run a Calculation Script, and Copy Data from a Block Storage Database to an Aggregate Storage Database

This scenario shows how to use the EPM Cloud REST APIs to import data, run a calculation script, and copy data from a block storage database to an aggregate storage database.

Example 4-4 Java

```
public void integrationScenarioImportDataRunCalcCopyToAso() throws
Exception {
    uploadFile("data.csv");
    executeJob("IMPORT_DATA", "loadingq1data",
"{importFileName:data.csv}");
    executeJob("CUBE_REFRESH", null, null);
    executeJob("PLAN_TYPE_MAP", "CampaignToReporting",
"{clearData:false}");
}
```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: see [Java Sample – UploadFile.java](#) and [Java Sample – ExecuteJob.java](#) in [Upload and Download Files](#).

Example 4-5 cURL

```
funcIntegrationScenarioImportDataRunCalcCopyToAso() {
    funcUploadFile "data.csv"
    funcExecuteJob "IMPORT_DATA" "loadingqldata" "{importFileName=data.csv}"
    funcExecuteJob "CUBE_REFRESH","cubeRefresh"
    funcExecuteJob "PLAN_TYPE_MAP" "CampaignToReporting" "{clearData=false}"
}
```

Example 4-6 Groovy

```
def integrationScenarioImportDataRunCalcCopyToAso() {
    uploadFile("data.csv");
    executeJob("IMPORT_DATA", "loadingqldata", "importFileName=data.csv");
    executeJob("CUBE_REFRESH", "cubeRefresh", null);
    executeJob("PLAN_TYPE_MAP", "CampaignToReporting", "clearData=false");
}
```

Scenario 3: Export and Download Metadata and Data

This scenario shows how to use the EPM Cloud REST APIs to export and download metadata and data.

Example 4-7 Java

```
public void integrationScenarioExportMetadataAndDataAndDownloadFiles()
throws Exception {
    executeJob("EXPORT_METADATA", "exportentitymetadata",
"{exportZipFileName:entitydata.zip}");
    executeJob("EXPORT_DATA", "Forecastdata",
"{exportFileName:forecastdata.zip}");
    listFiles();
    downloadFile("entitydata.zip");
    downloadFile("forecastdata.zip");
}
```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: see [Java Sample – DownloadFile.java](#) and [Java Sample – ExecuteJob.java](#) in [Upload and Download Files](#).

Example 4-8 cURL

```
funcIntegrationScenarioExportMetadataAndDataAndDownloadFiles() {
    funcExecuteJob "EXPORT_METADATA" "exportentitymetadata"
"{exportZipFileName=entitydata.zip}"
    funcExecuteJob "EXPORT_DATA" "Forecastdata"
"{exportFileName=forecastdata.zip}"
    funcListFiles
    funcDownloadFile "entitydata.zip"
```

```

    funcDownloadFile "forecastdata.zip"
}

```

Example 4-9 Groovy

```

def integrationScenarioExportMetadataAndDataAndDownloadFiles() {
    executeJob("EXPORT_METADATA", "exportentitymetadata",
"exportZipFileName=entitydata.zip");
    executeJob("EXPORT_DATA", "Forecastdata",
"exportFileName=forecastdata.zip");
    listFiles();
    downloadFile("entitydata.zip");
    downloadFile("forecastdata.zip");
}

```

Scenario 4: Remove Unnecessary Files from a Service Instance

This scenario shows how to use the EPM Cloud REST APIs to remove unnecessary files from a service instance.

Example 4-10 Java

```

public void integrationScenarioRemoveUnnecessaryFiles() throws
Exception {
    listFiles();
    deleteFile("entitymetadata.csv");
    deleteFile("forecastdata.csv");
}

```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: See [Java Sample — ListFiles.java](#) and [Java Sample — DeleteFile.java](#) in [View and Delete Files](#).

Example 4-11 cURL

```

funcIntegrationScenarioRemoveUnnecessaryFiles() {
    funcListFiles
    funcDeleteFile "entitymetadata.csv"
    funcDeleteFile "forecastdata.csv"
}

```

Example 4-12 Groovy

```

def integrationScenarioRemoveUnnecessaryFiles() {
    listFiles();
    deleteFile("entitymetadata.csv");
    deleteFile("forecastdata.csv");
}

```

Scenario 5: Archive Backups from the Service to Onpremise

This scenario shows how to use the EPM Cloud REST APIs to archive backups from the service to onpremise.

Example 4-13 Java

```
public void integrationScenarioExportDataAndDownloadFiles() throws Exception
{
    executeJob("EXPORT_DATA", "entitydata",
"{exportFileName:entitydata.zip}");
    executeJob("EXPORT_DATA", "forecastdata",
"{exportFileName:forecastdata.zip}");
    listFiles();
    downloadFile("entitydata.zip");
    downloadFile("forecastdata.zip");
}
```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: See Java Sample — ExecuteJob.java and Java Sample — DownloadFile.java in [Upload and Download Files](#).

Example 4-14 cURL

```
funcIntegrationScenarioExportDataAndDownloadFiles() {
    funcExecuteJob "EXPORT_DATA" "entitydata"
"{exportFileName:entitydata.zip}"
    funcExecuteJob "EXPORT_DATA" "forecastdata"
"{exportFileName:forecastdata.zip}"
    funcListFiles
    funcDownloadFile "entitydata.zip"
    funcDownloadFile "forecastdata.zip"
}
```

Example 4-15 Groovy

```
def integrationScenarioExportDataAndDownloadFiles() {
    executeJob("EXPORT_DATA", "entitydata", "exportFileName:entitydata.zip");
    executeJob("EXPORT_DATA", "forecastdata",
"exportFileName:forecastdata.zip");
    listFiles();
    downloadFile("entitydata.zip");
    downloadFile("forecastdata.zip");
}
```

Scenario 6: Refreshing the Application

This scenario shows how to use the EPM Cloud REST APIs to refresh the application.

Example 4-16 Java

```
public void integrationScenarioRefreshTheApplication() throws
Exception {
    uploadFile("accounts.zip");
    executeJob("IMPORT_METADATA", "accountMetadata",
"{importZipFileName:accounts.zip}");
    executeJob("CUBE_REFRESH", null, null);
}
```

Common Functions: See [Common Helper Functions for Java](#).

Dependent APIs: See [Java Sample — ExecuteJob.java](#) and [Java Sample — UploadFile.java](#) in [Upload and Download Files](#).

Example 4-17 cURL

```
funcIntegrationScenarioRefreshTheApplication() {
    funcUploadFile "accounts.zip"
    funcExecuteJob "IMPORT_METADATA" "accountMetadata"
"{importZipFileName:accounts.zip}"
    funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}
```

Example 4-18 Groovy

```
def integrationScenarioRefreshTheApplication() {
    uploadFile("accounts.zip");
    executeJob("IMPORT_METADATA", "accountMetadata",
"importZipFileName:accounts.zip");
    executeJob("CUBE_REFRESH", "cubeRefresh", null);
}
```

Scenario 7: Cloning an Instance

This scenario shows how to use the EPM Cloud REST APIs to clone an instance.

There are three ways to clone an environment. For this scenario, use one of the following procedures:

- Use [Clone Environment user interface](#)
- Use [EPM Automate](#)
- Use [a REST API](#)

Scenario 8: Sample Starter Kit for Consultants - Business Intelligence Cloud Integration

This scenario provides a sample starter kit for consultants to integrate with Oracle Business Intelligence Cloud.

A sample starter kit can be used by infrastructure consultants to plan integration for Planning with Business Intelligence Cloud.

Prerequisites

- You have accounts for Business Intelligence Cloud , Planning, and Oracle Application Express.
- You have considerable technical and functional expertise with Business Intelligence Cloud, Planning, Oracle Application Express, REST, Groovy, and scripting.

For detailed information, see [Sample Starter Kit for Consultants - Integration with Business Intelligence Cloud Service](#).

Scenario 9: Using Groovy Business Rules to Call REST APIs from Oracle and Other Companies

The scenario shows how to use the Oracle Enterprise Performance Management Cloud Groovy object model to call Oracle REST APIs and REST APIs developed by other companies.

These tutorials show you how to call a Data Management REST API to execute a data load rule and how to call the Google Places REST API from a Groovy script to add or update employee address information in Planning.

To learn how to use Groovy business rules to call Oracle and external REST APIs:

- [Calling a REST API from Oracle Using Groovy](#)
- [Calling a REST API from Other Companies Using Groovy](#)

To get an introduction to Groovy business rules:

- [Learning Groovy in Oracle EPM Cloud video](#)
- [Creating a Groovy Business Rule](#) in *Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud*
- [Introduction to Groovy Business Rules tutorial](#)
- [Additional Groovy Tutorials](#)
- [EPM Cloud Groovy Rules Java API Reference](#)

5

Quick Reference Table – REST API Resource View

The REST resources provide powerful APIs that you can use to manage Oracle Enterprise Performance Management Cloud as an alternative to using the web-based user interface.

The following table summarizes the REST resource paths.

Table 5-1 Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
EPM		
<i>/HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections/{connectionRef}</i>	GET	View a Connection
<i>/HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections</i>	GET	View All Connections
<i>/HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections/{connectionRef}</i>	GET	Update a Connection
Planning, FreeForm, Strategic Workforce Planning, and Sales Planning		
<i>/HyperionPlanning/rest/</i>	GET	Getting API Versions for Planning
<i>/HyperionPlanning/rest/{api_version}</i>	GET	Get Information about a Specific REST API Version for Planning
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/jobdefinitions</i>	GET	Get Job Definitions
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/jobs</i>	POST	Execute a Job
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/jobs/{jobIdentifier}</i>	GET	Retrieve Job Status
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/jobs/{jobIdentifier}/details</i>	GET	Retrieve Job Status Details
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/jobs/{jobIdentifier}/childjobs/{childJobIdentifier}/details</i>	GET	Retrieve Child Job Status Details
<i>/HyperionPlanning/rest/{api_version}/applications/{application}/dimensions/{dimname}/members</i>	POST	Add Member

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/HyperionPlanning/rest/{api_version}/applications/{application}/dimensions/{dimname}/members/{member}	GET	Get Member
/HyperionPlanning/rest/{api_version}/applications	GET	Get Applications
/HyperionPlanning/rest/{version}/applications/{application}/planningunits? q={"scenario":"scenarioName","version":"versionName"}&offset=10&limit=10	POST	List All Planning Units
/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits?q={"scenario": {"scenario"},"version": {"version"}}&offset={offset}&limit={limit}	GET	Get Planning Unit History and Annotations
/HyperionPlanning/rest/{api_version}/applications/{application}/users/{userId}/photo	GET	Get a Planning Unit Owner Photo
/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits/{puIdentifier}/promotionpath	GET	Get Planning Unit Promotional Path
/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits{puIdentifier}/availableactions	POST	Get Available Planning Unit Actions
/HyperionPlanning/rest/{api_version}/applications/{application}/pufilters	GET	Get Filters with All Possible Values
/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits/{puIdentifier}/actions	POST	Change Planning Unit Status
/HyperionPlanning/rest/{api_version}/applications/{application}/userpreferences	GET	Get User Preferences
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/importdataslice	POST	Import Data Slice
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/exportdataslice	POST	Export Data Slice
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/cleardataslice	POST	Clear Data Slice
/HyperionPlanning/rest/{api_version}/applications/{application}/substitutionvariables	GET	Get All Substitution Variables Defined for the Application

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/HyperionPlanning/rest/{api_version}/applications/{application}/substitutionvariables/MyPeriod	GET	Get a Substitution Variable Defined for the Application
/HyperionPlanning/rest/{api_version}/applications/{application}/substitutionvariables	POST	Create or Update All Substitution Variables Defined for the Application
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables	GET	Get Substitution Variables Defined at the Plan Type Level
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables?q={"derivedValues":true}	GET	Get Derived Substitution Variables at the Plan Type Level
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables/CurrYear	GET	Get a Substitution Variable Defined at the Plan Type Level
HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables/MyPeriod?q={"derivedValues":true}	GET	Get a Derived Substitution Variable Defined at the Plan Type Level
/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables	POST	Create and Update Substitution Variables at the Plan Type Level
Migration		
/interop/rest/	GET	Get REST API Versions for Migration
/interop/rest/{api_version}	GET	Get Information about a Specific Version of Migration Sample Code
/interop/rest/11.1.2.3.600/applicationsnapshots/{applicationSnapshotName}/contents	POST	Upload
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents	GET	Download
/interop/rest/{api_version}/applicationsnapshots	GET	List Files (v11.1.2.3.600)
/interop/rest/v2/files/list	GET	List Files (v2)
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}	DELETE	Delete Files (v11.1.2.3.600)
/interop/rest/v2/files/delete	DELETE	Delete Files (v2)
/interop/rest/v3/files/delete		Delete Files (v3)
/interop/rest/{api_version}/services	GET	Get Information About All Services
/interop/rest/{api_version}/services/{servicename}/recreate	POST	Run Recreate on a Service (11.1.2.3.600)
/interop/rest/v2/config/services/recreate	POST	Run Recreate on a Service (v2)

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/interop/rest/{api_version}/services/{service_type}/resetservice	POST	Restart the Service Instance (v1)
/interop/rest/v2/config/services/reset	POST	Restart the Service Instance (v2)
/interop/rest/{api_version}/applicationsnapshots	GET	Get Information About All Application Snapshots
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}	GET	Get Information about a Specific Application Snapshot Sample Code
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents?q={"isLast":false,"isFirst":true,"chunkSize":14,"fileSize":55445}	POST	Upload Application Snapshot
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/content	GET	Download Application Snapshot
/interop/rest/{api_version}/feedback	POST	Provide Feedback (v11.1.2.3.600)
/interop/rest/v2/services/feedback	POST	Provide Feedback (v2)
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/migrationq={type:"export"}	POST	LCM Export (v1)
/interop/rest/v2/snapshots/export	POST	LCM Export (v2)
/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/migration?q={type:"import"}	POST	LCM Import (v1)
/interop/rest/v2/snapshots/import	POST	LCM Import (v2)
/interop/rest/v1/services/skipupdate	POST	Skip Updates (v1)
/interop/rest/v2/services/skipupdate	POST	Skip Updates (v2)
/interop/rest/<api_version>/services/sendmail	POST	Send Email (v1)
/interop/rest/v2/emails/send	POST	Send Email (v2)
/interop/rest/{api_version}/services/dailymaintenance	GET	Get the Build Version and Daily Maintenance Time (v1)
/interop/rest/v2/maintenance/getdailymaintenancestarttime	GET	Get the Build Version and Daily Maintenance Window Time (v2)
/interop/rest/{api_version}/services/dailymaintenance?StartTime={N}	PUT	Setting the Daily Maintenance Time (v1)
/interop/rest/v2/maintenance/setdailymaintenancestarttime	PUT	Setting the Daily Maintenance Time (v2)
/interop/rest/{api_version}/services/maintenancewindow	POST	Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v1)
/interop/rest/v2/maintenance/rundailymaintenance	POST	Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/interop/rest/v1/services/{servicename}/ copysnapshot	POST	Copy Application Snapshot (v1)
/interop/rest/v2/snapshots/ copyfrominstance	POST	Copy Application Snapshot (v2)
/interop/rest/v1/renamesnapshot	PUT	Rename Application Snapshot (v1)
/interop/rest/v2/snapshots/rename	PUT	Rename Application Snapshot (v2)
/interop/rest/v1/services/ copytoobjectstore	POST	Copy to Object Store (v1)
/interop/rest/v2/objectstorage/copyto	POST	Copy to Object Store (v2)
/interop/rest/v1/services/ copyfromobjectstore	POST	Copy from Object Store (v1)
/interop/rest/v2/objectstorage/copyfrom	POST	Copy from Object Store (v2)
/interop/rest/security/<api_version>/ users	POST	Add Users to an Identity Domain
/interop/rest/security/<api_version>/ users?filename=<filename>	DELETE	Remove Users from an Identity Domain
/interop/rest/security/<api_version>/ users	PUT	Assign Users to Predefined or Application Roles
/interop/rest/security/<api_version>/ users	PUT	Remove Users' Role Assignment
/interop/rest/security/<api_version>/ users	POST	Add Users to an Identity Domain
/interop/rest/security/<api_version>/ users?filename=<filename>	DELETE	Remove Users from an Identity Domain
/interop/rest/security/<api_verion>/ users	PUT	Update Users
/interop/rest/{api_version}/reports? q={type:userauditreport,fileName:useraudit report.csv,since:2017-12-10,until:2018-0 6-10}	POST	User Audit Report (v1)
/interop/rest/v2/reports/useraudit	POST	User Audit Report (v2)
/interop/rest/security/<api_version>/ groups	PUT	Add a User to a Batch of Groups
/interop/rest/security/<api_version>/ groups	PUT	Remove a User from a Batch of Groups
/interop/rest/security/<api_version>/ groups	POST	Add a Batch of Groups
/interop/rest/security/<api_version>/ groups	DELETE	Remove a Batch of Groups
/interop/rest/security/<api_version>/ usergroupreport	POST	User Group Report
/interop/rest/{api_version}/reports? q={type:provisionreport,fileName:provrep ort.csv,format:simplified,usertype,servi ceusers}	POST	User Access Report (v1)

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/interop/rest/v2/reports/useraccess	POST	User Access Report (v2)
/interop/rest/security/{api_version}/roleassignmentreport	POST	Role Assignment Report
/interop/rest/security/{api_version}/roleassignmentauditreport/	POST	Role Assignment Audit Report for OCI (Gen 2) Environments
/interop/rest/security/{api_version}/invalidloginreport/	POST	Invalid Login Report for OCI (Gen 2) Environments
/interop/rest/epmociservice/v2/ipallowlist	GET	View the IP Allowlist - Only for OCI (Gen 2) Environments
/interop/rest/epmociservice/v2/ipallowlist	POST	Update the IP Allowlist - Only for OCI (Gen 2) Environments
/interop/restp{api_version}/reports/groupaudit	POST	Group Assignment Audit Report
/interop/rest/{api_version}/services/encryptionkey	PUT	Set Encryption Key (v1)
/interop/rest/v2/services/setencryptionkey	PUT	Set Encryption Key (v2)
/interop/rest/security/<api_version>/groups	PUT	Add a User To a Batch of Groups
/interop/rest/security/<api_version>/groups	PUT	Remove a User from a Batch of Groups
/interop/rest/v1/services/copyfile	POST	Copy a File Between Instances (v1)
/interop/rest/v2/files/copyfrominstance	POST	Copy a File Between Instances (v2)
/interop/rest/v1/services/clone	POST	Clone an Environment
/interop/rest/v2/backups/list	GET	List Backups - Only for OCI (Gen 2) Environments
/interop/rest/v2/backups/restore	POST	Restore Backup - Only for OCI (Gen 2) Environments
/interop/rest/v2/essbase/export	POST	Export Essbase Data (v2)
Data Management		
/aif/rest/	GET	Get API Versions for Data Management APIs
/aif/rest/{api_version}	GET	Get Information about a Specific API Version for Data Management APIs
/aif/rest/V1/POV	POST	Lock and Unlock POV
/aif/rest/{api_version}/jobs	POST	Running Data Rules
/aif/rest/{api_version}/jobs	POST	Integration Job Type
/aif/rest/{api_version}/jobs	POST	Running Batch Rules
/aif/rest/{api_version}/jobs	POST	Import Data Mapping
/aif/rest/{api_version}/jobs	POST	Export Data Mapping
/aif/rest/{api_version}/jobs	POST	Execute Reports for Data Management
/aif/rest/V1/snapshots	POST	Execute Reports for Data Management
Account Reconciliation		
/armARCS/rest/	GET	Get API Versions for Account Reconciliation REST APIs

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/armARCS/rest/<api_version>	GET	Get Information about a Specific API Version for Account Reconciliation REST APIs
/armARCS/rest/{api_version}/jobs	POST	Execute a Job in Account Reconciliation
/armARCS/rest/{version}/jobs	POST	Create Reconciliation (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Change Period Status (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Pre-Mapped Transactions (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Profiles (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Rates (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Balances (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Pre-Mapped Balances (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Import Pre-Mapped Transactions (Transaction Matching)
/armARCS/rest/{version}/jobs	POST	Import Reconciliation Attributes (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Monitor Reconciliations (Reconciliation Compliance)
/armARCS/rest/{version}/jobs	POST	Run Auto Match (Transaction Matching)
/armARCS/rest/{version}/jobs	POST	Adding Users to a Team for Account Reconciliation
/armARCS/rest/{version}/jobs	POST	Removing Users from a Team for Account Reconciliation
/armARCS/rest/{api_version}/jobs/{job_id}	GET	Retrieve Job Status for Account Reconciliation
/arm/rest/fcmapi/{api_version}/report	POST	Generate Report for Account Reconciliation
/arm/rest/fcmapi/{api_version}/rc/export/users	POST	Generate User Details Report for Financial Consolidation and Close and Tax Reporting
/arm/rest/fcmapi/{api_version}/rc/export/applicationproperties	POST	Export Application Properties
/arm/rest/fcmapi/{api_version}/rc/import/applicationproperties	POST	Import Application Properties
/arm/rest/fcmapi/{api_version}/rc/export/backgroundImage	POST	Export Background Image
/arm/rest/fcmapi/{api_version}/rc/import/backgroundImage	POST	Import Background Image
/arm/rest/fcmapi/{api_version}/rc/export/logo	POST	Export Logo Image
/arm/rest/fcmapi/{api_version}/rc/import/logo	POST	Import Logo Image
/arm/rest/fcmapi/{api_version}/{module}/connections	POST	Create a Connection
/arm/rest/fcmapi/{api_version}/{module}/connections	GET	View All Connections

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/arm/rest/fcmapi/{api_version}/{module}/connections/{id}	PUT	Update a Connection
/arm/rest/fcmapi/{api_version}/{module}/connections/{id}	DELETE	Delete a Connection
POST /armARCS/rest/{api_version}/appaccess		Set Application Access Level
GET /armARCS/rest/{api_version}/appaccess		Retrieve Application Access Level
Financial Consolidation and Close		
/HyperionPlanning/rest/	GET	Getting API Versions for Financial Consolidation and Close
/fccs/rest/{api_version}	GET	Get Information about a Specific API Version for Financial Consolidation and Close APIs
/HyperionPlanning/rest/cmapi/{api_version}/jobs	POST	Deploy Task Manager Templates
/HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs	POST	Import Supplemental Collection Data for Financial Consolidation and Close
/HyperionPlanning/rest/{api_version}/applications/{application}/journals/{journal}/actions	POST	Perform Journal Actions for Financial Consolidation and Close
/HyperionPlanning/rest/{api_version}/applications/{application}/journalPeriods/{period}/actions	POST	Perform Journal Period Updates for Financial Consolidation and Close
/HyperionPlanning/rest/{api_version}/applications/{application}/journals?q={"scenario","year","period","status"}&offset=0&limit=5	GET	Retrieve Journals for Financial Consolidation and Close
/HyperionPlanning/rest/ej/{api_version}/jobs	POST	Execute an Enterprise Journals Job
/HyperionPlanning/rest/{api_version}/applications/{application}/jobs	POST	Copy Data
/HyperionPlanning/rest/{api_version}/applications/{application}/jobs	POST	Clear Data
/HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs	POST	Deploy Form Templates
/HyperionPlanning/rest/cmapi/{api_version}/updateTasksForEventMonitoring	POST	Update Task Status for Event Monitoring
/interop/rest/{api_version}/services/dataaccess?accessType={allow revoke}&disableEmergencyAccess={true false}	PUT	Manage Permission for Manual Access to Database (v1)
/interop/rest/v2/services/setmanualdataaccess	PUT	Manage Permission for Manual Access to Database (v2)
/HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs	POST	Adding Users to a Team for Financial Consolidation and Close and Tax Reporting

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs	POST	Removing Users from a Team for Financial Consolidation and Close and Tax Reporting
/HyperionPlanning/rest/{api_version}/applications/{application}/journals/{journal label}?q={"scenario","year","period"}&lineItems=true	GET	Retrieve Journal Details for Financial Consolidation and Close
/HyperionPlanning/rest/ej/v1/ejjournals	GET	Retrieve Enterprise Journals for Financial Consolidation and Close
/HyperionPlanning/rest/ej/v1/ejjournals/{instanceId}	GET	Retrieve Enterprise Journal Content for Financial Consolidation and Close
/HyperionPlanning/rest/ej/v1/ejjournalcontent?q={"year","period"}	GET	Retrieve Enterprise Journal Content by Year and Period for Financial Consolidation and Close
/HyperionPlanning/rest/ej/v1/ejjournals/{instanceId}/poststatus	POST	Update Enterprise Journal Posting Status for Financial Consolidation and Close
/HyperionPlanning/rest/fcmapi/{api_version}/report	POST	Generate Report for Financial Consolidation and Close and Tax Reporting
/HyperionPlanning/rest/fcmapi/{api_version}/fcm/export/users	POST	Generate User Details Report for Financial Consolidation and Close and Tax Reporting
Enterprise Profitability and Cost Management		
/epm/rest/	GET	Getting API Versions for Enterprise Profitability and Cost Management
/epm/rest/{api_version}	GET	Getting Information About a Specific REST API Version for Enterprise Profitability and Cost Management
/HyperionPlanning/rest/v3/applications/{AppName}/jobs/	POST	Calculate Model
/HyperionPlanning/rest/v3/applications/{AppName}/jobs/	POST	Clear Data By Point Of View
/HyperionPlanning/rest/v3/applications/{AppName}/jobs/	POST	Copy Data by Point of View
/HyperionPlanning/rest/v3/applications/{AppName}/jobs/	POST	Delete Point of View
/HyperionPlanning/rest/v3/applications/{AppName}/jobs/	POST	Validate Model
Profitability and Cost Management		
/epm/rest/	GET	Get API Versions for Profitability and Cost Management REST APIs
/epm/rest/{api_version}	GET	Get Information about a Specific API Version for Profitability and Cost Management
/epm/rest/{api_version}/applications/jobs/ChecktaskStatusJob/{processName}	GET	Retrieve Task Status for Profitability and Cost Management
/epm/rest/{api_version}/applications/{application}/jobs/ledgerDeployCubeJob	POST	Deploy ML Cube
/epm/rest/{api_version}/applications/{application}/jobs/mergeSlices	POST	Merge Slices for Profitability and Cost Management

Table 5-1 (Cont.) Quick Reference Table - REST API Resource View

REST Resource	Request	More Information
/epm/rest/{api_version}/applications/{application}/povs/{povGroupMember}/jobs/runLedgerCalculationJob	POST	Run ML Calculations
/epm/rest/{api_version}/applications/{application}/povs/{povGroupMember}/jobs/runLedgerCalculationJob	POST	Run ML Clear POV
/epm/rest/{api_version}/applications/{application}/povs/{srcPOVMemberGroup}/jobs/copyPOVJob/{destPOVMemberGroup}	POST	Copy ML POV Data
/epm/rest/{api_version}/applications/{application}/povs/{povGroupMember}/jobs/runLedgerCalculationJob	POST	Run ML Rule Balancing
/epm/rest/{api_version}/applications/{application}/jobs/essbaseDataLoadJob	POST	Essbase Data Load for Profitability and Cost Management
/epm/rest/v1/fileApplications/{applicationName}	POST	Create File-Based Application
/epm/rest/{api_version}/fileApplications/{application}/updateDimension	POST	Update File-Based Application
/epm/rest/{api_version}/fileApplications/{application}/enableApplication	POST	Enable File-Based Application
/epm/rest/{api_version}/applications/{application}/templateExportJob?fileName={fileName}	POST	Export Template for Profitability and Cost Management
/epm/rest/{api_version}/applications/{application}/templateImportJob	POST	Import Template for Profitability and Cost Management
/epm/rest/{api_version}/applications/{application}/exportQueryResultsJob	POST	Export Query Results
/epm/rest/{api_version}/applications/{application}/jobs/applyDataGrants	POST	Apply Data Grants
/epm/rest/{api_version}/applications/{application}/povs/{POV}/programDocumentationReport?queryParameter={"fileType":"PDF","useAlias":"true"}	GET	Generate Program Documentation Report
/epm/rest/{api_version}/applications/<applicationName>/<povName>/jobs/programDocReportJob	POST	Generate Program Documentation Report - Run as a Job
/epm/rest/v1/applications/{AppName}/jobs/optimizeASOCube	POST	Optimize ASO Cube

6

REST Resources and Methods

This section describes the REST APIs for Oracle Enterprise Performance Management Cloud.

Completing administrative tasks using REST APIs as an alternative to using the user interface requires considerable technical and functional expertise. Only technically competent EPM Cloud Administrators should use this guide to perform EPM Cloud Administrator administrative tasks. For prerequisites to using these REST APIs, see [Prerequisites](#).

The predefined and application roles assigned to the user of the REST API determine which APIs can be executed.

Supported REST Methods

You can use the Oracle Enterprise Performance Management Cloud REST APIs to create and manage resources for selected functionality. These APIs provide an alternative to using the selected components in the web-based user interface.

You can use one of a variety of methods to access the REST APIs. For example, you can access the REST API through client applications such as:

- Web browsers
- cURL

You can also use the REST APIs in REST client applications that are developed in languages such as:

- JavaScript
- Ruby
- Perl
- Java
- Groovy

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST describes any simple interface that transmits data over a standardized interface (such as HTTP) without an additional messaging layer, such as SOAP. REST provides a set of design rules for creating stateless services that are viewed as resources, or sources of specific information, and can be identified by their unique URIs. RESTful web services are services that are built according to REST principles and, as such, are designed to work well on the web. Typically, RESTful web services are built on the HTTP protocol and implement operations that map to the common HTTP methods, such as GET, POST, PUT, and DELETE to retrieve, create, update, and delete resources, respectively.

REST API Methods

You can create, view, update, or delete Oracle Enterprise Performance Management Cloud resources using standard HTTP method requests, as summarized in the following table.

Table 6-1 REST API Methods

Method	Description
GET	Retrieve information about the REST API resource
POST	Create a REST API resource
PUT	Update a REST API resource
DELETE	Delete a REST API resource or related component

Error Handling

All REST APIs return JSON output appropriate for the API invoked. HTTP Status codes other than 200 are used as appropriate to indicate various failures, along with JSON for detailed error messages.

Versioning

The Oracle Enterprise Performance Management Cloud REST API web services are versioned at the API level and expect the version to be included in the URL as shown here. An error will occur if the API version is missing or the provided version is not supported by the API.

For each service's API, you can get version and details for a specific REST API version. For details, see the service's API topics or [Current REST API Version](#).

Important: The version number is case-sensitive. For example, if the version number is listed as `v1` with a lowercase `v`, you cannot enter the version number with a capital `V` as in this incorrect example, `V1`, which would result in an error. Instead, you must enter the version number with a lowercase `v` as in this correct example: `v1`.

Examples:

```
https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/  
rest/{api_version}/applications/{applicationName}/jobs
```

```
https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/  
{api_version}/applicationsnapshots
```

where `{api_version}` is the current REST API version for the product, for example, `v3` for Planning.

Current REST API Version

Use the *Getting API Versions* topic for the Oracle Enterprise Performance Management Cloud business process to get the API version number and details for a specific REST API.

- Planning: See [Getting API Versions for Planning](#)
- Migration: See [Getting API Versions for Migration APIs](#)
- Data Management: See [Getting API Versions for Data Management APIs](#)
- Account Reconciliation: See [Getting API Versions for Account Reconciliation REST APIs](#)
- Financial Consolidation and Close: See [Getting API Versions for Financial Consolidation and Close APIs](#)
- Profitability and Cost Management: See [Getting API Versions for Profitability and Cost Management REST APIs](#)
- Profitability and Cost Management: See [Getting API Versions for Enterprise Profitability and Cost Management](#)

Status Codes

When you call any of the Oracle Enterprise Performance Management Cloud REST APIs, one of the following standard HTTP status codes is returned in the response header.

Table 6-2 Status Codes

HTTP Status Codes	Description
200 OK	The request was successfully completed. A 200 status is returned for a successful GET or POST method.
201 Created	The request has been fulfilled and resulted in a new resource being created. The response includes a Location header containing the canonical URI for the newly created resource. A 201 status is returned from a synchronous resource creation or an asynchronous resource creation that completed before the response was returned.
202 Accepted	The request has been accepted for processing, but the processing has not been completed. The request may or may not eventually be acted upon, as it may be disallowed at the time processing actually takes place. When specifying an asynchronous (<code>_detached=true_</code>) resource creation, for example, when deploying an application, or update, for example, when redeploying an application, a 202 is returned if the underlying operation does not complete in a reasonable amount of time. The response contains a Location header of a job resource that the client should poll to determine when the job has finished. It also returns an entity that contains the current state of the job.
400 Bad Request	The request could not be processed because it contains missing or invalid information, such as a validation error on an input field, a missing required value, and so on.
401 Unauthorized	The request is not authorized. The authentication credentials included with this request are missing or invalid.
403 Forbidden	The user cannot be authenticated. The user does not have authorization to perform this request.

Table 6-2 (Cont.) Status Codes

HTTP Status Codes	Description
404 Not Found	The request includes a resource URI that does not exist
405 Method Not Allowed	The HTTP verb specified in the request (DELETE, GET, POST, PUT) is not supported for this request URI.
406 Not Acceptable	The resource identified by this request is not capable of generating a representation corresponding to one of the media types in the Accept header of the request.
415 Not Acceptable	The client's ContentType header is not correct.
500 Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
503 Service Unavailable	The server is unable to handle the request due to temporary overloading or maintenance of the server.

7

Planning REST APIs

Use the Planning REST APIs to get the REST API version, manage and execute jobs, and work with members, applications, planning units, user preferences, data slices, and substitution variables.

Note: We have removed the following fields from the exception response in REST APIs for Planning and Planning modules:

- message
- localizedMessage

URL Structure for Planning

This topic shows the general URL structure for Planning REST APIs.

Use the following URL structure to access the Planning REST resources:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/{api_version}/{path}
```

Where:

api_version —API version you are developing with. The current REST API version for Planning is v3.

path —Identifies the resource

Resources and Available Actions

In the response, the Links section of the response parameters lists links to other resources and available actions for the current resource.

Table 7-1 Resources and Available Actions

Name	Description
links	Describes links to other resources and actions applicable on the current resource
rel	Relationship type; the relationship between the current state and the state to which the client will transition
href	The target resource's URI. If the value of rel is "self", this URI is how the resource is accessed currently
action	The HTTP method. For POST, data indicates the parameters and values with which it was invoked

Getting API Versions for Planning

You can get information on REST API versions using a set of REST resources, as summarized here.

Important: The version number is case-sensitive. For example, if the version number is listed as `v3` with a lowercase `v`, you cannot enter the version number with a capital `V` as in this incorrect example, `V3`, which would result in an error. Instead, you must enter the version number with a lowercase `v` as in this correct example: `v3`.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 7-2 Getting REST API Versions

Task	Req uest	REST Resource
Get REST API Versions for Planning	GET	/HyperionPlanning/rest/
Get Information about a Specific REST API Version for Planning	GET	/HyperionPlanning/rest/ {api_version}

Get REST API Versions for Planning

Returns information about which versions are available and supported. Multiple versions might be supported simultaneously by Planning.



Note:

An API version is always supported even when deprecated.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /HyperionPlanning/rest/

Request

Supported Media Types: application/json

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-3 Parameters

Name	Description
items	Version of the API you are developing with
version	The version, such as v3
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "version": "v1",
    "lifecycle": "deprecated",
    "isLatest": false,
    "links": [{
      "rel": "canonical",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v1"
    }, {
      "rel": "successor-version",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v2"
    }
  ]
}, {
  "version": "v2",
  "lifecycle": "deprecated",
  "isLatest": false,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v2"
  }, {
    "rel": "predecessor-version",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v1"
  }, {
    "rel": "successor-version",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3"
  }
  ]
}, {
  "version": "v3",
  "lifecycle": "active",
  "isLatest": true,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3"
  }, {
```



```

        "rel": "predecessor-version",
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v2"
    }
  }
},
"links": [{
  "rel": "self",
  "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/"
}, {
  "rel": "canonical",
  "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/"
}, {
  "rel": "current",
  "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3"
}
}
}

```

Get Information about a Specific REST API Version for Planning

Returns information about a specific REST API version for Planning.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /HyperionPlanning/rest/{api_version}

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-4 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V3	Path	Yes	None

Response Body

Supported Media Types: application/json

Parameters

The following table summarizes the response parameters.

Table 7-5 Parameters

Attribute	Description
version	The version, such as v3
lifecycle	Lifecycle of the resource, active or deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "version": "v3",
  "lifecycle": "active",
  "isLatest": true,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3"
  }, {
    "rel": "predecessor-version",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v2"
  }]
}
```

Manage Jobs

You can manage jobs using a set of REST resources, as summarized here.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Jobs:

- [Get Job Definitions](#)
- [Execute a Job](#)
- [Retrieve Job Status](#)
- [Retrieve Job Status Details](#)
- [Rules](#)
- [Ruleset](#)
- [Plan Type Map](#)
- [Import Data](#)
- [Export Data](#)

- [Export Metadata](#)
- [Import Metadata](#)
- [Cube Refresh](#)
- [Clear Cube](#)
- [Administration Mode](#)
- [Compact Cube](#)
- [Restructure Cube](#)
- [Merge Data Slices](#)
- [Optimize Aggregation](#)
- [Import Security](#)
- [Export Security](#)
- [Export Audit](#)
- [Export Job Console](#)
- [Sort Members](#)
- [Import Exchange Rates](#)
- [Auto Predict](#)
- [Import Cell-Level Security](#)
- [Export Cell-Level Security](#)
- [Import Valid Intersections](#)
- [Export Valid Intersections](#)
- [Execute a Report Bursting Definition](#)

Get Job Definitions

Can be used to retrieve job definitions for types of jobs that can be scheduled to run.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/  
{application}/jobdefinitions
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-6 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
jobIdentifier	The ID of the job, such as 224	Path	Yes	None
q	Query string	Query	No	None
jobType	<p>Optionally, retrieve job definitions for a particular job type, such as RULES. These jobs are supported:</p> <ul style="list-style-type: none"> • Rules • Ruleset • Plan Type Map • Import Data • Export Data • Export Metadata • Import Metadata • Cube Refresh • Clear Cube • Administration Mode • Compact Cube • Restructure Cube • Merge Data Slices • Optimize Aggregation • Import Security • Export Security • Export Audit • Export Job Console • Sort Members • Import Exchange Rates • Auto Predict • Import Cell-Level Security • Export Cell-Level Security • Import Valid Intersections • Export Valid Intersections • Execute a Report Bursting Definition 			

Example URLs

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4app1/jobdefinitions
```

Specifying an optional jobType, RULES:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4app1/jobdefinitions?
q={"jobType":"RULES"}
```

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-7 Parameters

Name	Description
items	Collection of job definitions
jobType	Job type, such as RULESET. These jobs are supported: <ul style="list-style-type: none"> • Rules • Ruleset • Plan Type Map • Import Data • Export Data • Export Metadata • Import Metadata • Cube Refresh • Clear Cube • Administration Mode • Compact Cube • Restructure Cube • Merge Data Slices • Optimize Aggregation • Import Security • Export Security • Export Audit • Export Job Console • Sort Members • Import Exchange Rates • Auto Predict • Import Cell-Level Security • Export Cell-Level Security • Import Valid Intersections • Export Valid Intersections • Execute a Report Bursting Definition
jobName	The exact name of the job, such as Financial Statements - Forecast.
type	Application type

Example of Response Body

The following shows an example of the response body specifying jobType with a value of RULESET.

```
{
  "items": [{
    "jobType": "RULESET",
    "jobName": "Financial Statements - Forecast",
    "links": null
  }, {
    "jobType": "RULESET",
    "jobName": "Financial Statements - Plan",
    "links": null
  }, {
```

```

        "jobType": "RULESET",
        "jobName": "Revenue Forecast",
        "links": null
    }, {
        "jobType": "RULESET",
        "jobName": "Revenue Plan",
        "links": null
    }, {
        "jobType": "RULESET",
        "jobName": "RS 60 RTP vars test2",
        "links": null
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/vision/jobdefinitions?q=%7BjobType:RULESET%7D",
      "action": "GET"
    }
  ],
}

```

Execute a Job

Use this resource to execute several jobs simultaneously by providing the job name and type. The job is expected to be defined in Planning with all the required parameters saved with the job definition. For some job types, the parameters can be either provided or overwritten at runtime.

Supported job types

This topic describes general information for executing a job. Details for each job type are described in separate topics for individual jobs.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/jobs

Request

Supported Media Types: application/json

Parameters

This table summarizes the request parameters that are generic to all jobs. The following tables describe parameters specific to individual jobs.

Note that all parameter names and values are case sensitive.

Table 7-8 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which the job will be executed	Path	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/v3/applications/PS4appl/jobs
```

```
{"jobType":"jobType","jobName":"jobName","parameters":  
{"parameter1":"value","parameter2":"value2"}}
```

Response

Supported Media Types: application/json

Parameters

This table summarizes the response parameters that are generic to all jobs. The following topics describe parameters specific to individual jobs:

- [Rules](#)
- [Ruleset](#)
- [Plan Type Map](#)
- [Import Data](#)
- [Export Data](#)
- [Export Metadata](#)
- [Import Metadata](#)
- [Cube Refresh](#)
- [Clear Cube](#)
- [Administration Mode](#)
- [Compact Cube](#)
- [Restructure Cube](#)
- [Merge Data Slices](#)
- [Optimize Aggregation](#)
- [Import Security](#)
- [Export Security](#)
- [Export Audit](#)
- [Export Job Console](#)

- [Sort Members](#)
- [Import Exchange Rates](#)
- [Auto Predict](#)
- [Import Cell-Level Security](#)
- [Export Cell-Level Security](#)
- [Import Valid Intersections](#)
- [Export Valid Intersections](#)
- [Execute a Report Bursting Definition](#)

Table 7-9 Parameters

Name	Description
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 = cancel pending; 3 = cancelled; 4 = invalid parameter; Integer.MAX_VALUE = unknown
details	Details about the job status, such as "Metadata import was successful" for metadata import
jobID	The ID of the job, such as 145
jobName	The name of the job, such as Refresh Database.
descriptiveStatus	The status of the job, such as Completed or Error

Rules

Launches a business rule.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator, Power User (if Rule Launch access is granted)

REST Resource

```
POST      /HyperionPlanning/rest/{api_version}/applications/{application}/
jobs
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Table 7-10 Rules

Name	Description	Required	Default
jobType	Rules or RULES (both parameters are supported)	Yes	None
jobName	The name of a business rule exactly as it is defined in the Planning application. Example: RollupUSSales	Yes	None
parameters	Optionally you can specify the runtime prompts and their values required to execute the business rule. Note: The rule is executed against the plan type to which it was deployed. The value must use JSON syntax.	No, unless default values for the run time prompts are not provided in Calculation Manager.	Default values for the runtime prompts as provided in Calculation Manager will be used.

Example URL and Payload

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4app1/jobs`

```
{
  "jobType": "Rules",
  "jobName": "Operating Expense Adj Plan",
  "parameters":
    {
      "MyScenario1": "Current",
      "MyVersion1": "BU Version_1",
      "ToEntity": "CA",
      "Rule_Level_Var": "AZ",
      "planType": "Plan1"
    }
}
```

Ruleset

Launches a business ruleset.

Supports rulesets with no runtime prompts or runtime prompts with default values. You can add parameters to rulesets for greater flexibility. Use the sample rulesets and POST requests below to help you quickly understand different scenarios when running this job. For details about rulesets, see [Designing Business Rulesets](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#). For information about creating rulesets, see *Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud*.

Required Roles

Service Administrator, Power User (if Rule Launch access is granted)

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-11 Ruleset

Name	Description	Required	Default
jobType	Ruleset or RULESET (both parameters are supported).	Yes	None
jobName	The name of a business ruleset exactly as is defined in the Planning application. Example: RollupUSSales	Yes	None
parameters		No	Default values for the runtime prompts as provided in Calculation Manager will be used.

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/v3/applications/PS4appl/jobs
```

```
{
  "jobType": "Ruleset",
  "jobName": "Calculate Plan Operating Expenses"
}
```

Example Ruleset Scenarios and POST Requests

These examples based on the Vision application illustrate how to run rulesets with parameters. For each example, review the sample ruleset in Calculation Manager to understand the sample POST request.

Example 1: Ruleset "Revenue Plan" when variables are merged and not hidden

In Revenue Plan, by default the variables are merged and hidden. When variables are hidden, the default values defined in Calculation Manager will be used when executing the ruleset and any values provided in the payload will be ignored.

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Copy Channel	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>

Sample POST request

```
{
  "jobType": "Ruleset",
  "jobName": "Revenue Plan",
  "parameters": {
    "Version": "Worst Case",
    "Department": "No Entity"
  }
}
```

Example 2: Ruleset "Revenue Plan" when variables are not merged and not hidden

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Copy Channel	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>
Revenue Plan	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>

Sample POST request

```
{
  "jobType": "Ruleset",
  "jobName": "Revenue Plan",
  "parameters": {
    "Copy Channel.Department": "No Entity",
  }
}
```

```

"Copy Channel.Version":"Worst Case",

"Revenue Plan.Department":"New Entity",
"Revenue Plan.Version":"Best Case"
}
}

```

Example 3: Ruleset "Calculate Plan Operating Expenses" with nested ruleset ("Revenue Plan") and variables are merged and not hidden

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Calculate Plan Operating Expenses	Vision		
Operating Expenses Plan	Vision	Plan1	
Operating Expense Adj Plan	Vision	Plan1	
Copy Channel	Vision	Plan1	
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Operating Expenses Plan	Version	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
Copy Channel	Department	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>

Sample POST request

```

{
  "jobType":"Ruleset",
  "jobName":"Calculate Plan Operating Expenses",
  "parameters":
  {
    "Department":"Unspecified Entity",
    "Version":"Most Likely"
  }
}

```

Example 4: Ruleset "Calculate Plan Operating Expenses" with a nested ruleset ("Revenue Plan") and variables that are not merged and not hidden

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Calculate Plan Operating Expenses	Vision		
Operating Expenses Plan	Vision	Plan1	
Operating Expense Adj Plan	Vision	Plan1	
Copy Channel	Vision	Plan1	
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Operating Expenses Plan	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>
Operating Expense Adj Plan	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>
Copy Channel	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>
Copy Channel	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>
Revenue Plan	Department	Application	<input type="checkbox"/>		<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>		<input type="checkbox"/>

Sample POST request

In this example, the same rule, Copy Channel, appears twice – once under Calculate Plan Operating Expenses, and once under Revenue Plan. This demonstrates how to provide the variables with their fully qualified paths.

```
{
  "jobType": "Ruleset",
  "jobName": "Calculate Plan Operating Expenses",
  "parameters": {
    "Operating Expenses Plan.Version": "Most Likely",
    "Operating Expense Adj Plan.Version": "What If",

    "Copy Channel.Department": "Unspecified Entity",
    "Copy Channel.Version": "Working",

    "Revenue Plan.Copy Channel.Department": "New Entity",
    "Revenue Plan.Copy Channel.Version": "Best Case",

    "Revenue Plan.Revenue Plan.Department": "No Entity",
    "Revenue Plan.Revenue Plan.Version": "Worst Case"
  }
}
```

Optionally, you can provide the sequence indexes in the paths, as shown below.

```
{
  "jobType": "Ruleset",
  "jobName": "Calculate Plan Operating Expenses",
```

```

"parameters":
{
  "(1)Operating Expenses Plan.Version": "Most Likely",
  "(2)Operating Expense Adj Plan.Version": "What If",

  "(3)Copy Channel.Department": "Unspecified Entity",
  "(3)Copy Channel.Version": "Working",

  "(4.1)Revenue Plan.Copy Channel.Department": "New Entity",
  "(4.1)Revenue Plan.Copy Channel.Version": "Best Case",

  "(4.2)Revenue Plan.Revenue Plan.Department": "No Entity",
  "(4.2)Revenue Plan.Revenue Plan.Version": "Worst Case"
}
}

```

Example 5: Ruleset "Revenue Plan" with variables that are merged and not hidden

This example shows two Revenue Plan rules within a ruleset. This demonstrates how to differentiate the variables when there are multiple variables with the same paths within the same parent in the ruleset.

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Copy Channel	Department	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>

Sample POST request

```

{
  "jobType": "Ruleset",
  "jobName": "Revenue Plan",
  "parameters":
  {
    "Version": "Worst Case",
    "Department": "New Entity"
  }
}

```

Example 6: Ruleset "Revenue Plan" with variables that are not merged and not hidden

This example shows two Revenue Plan rules within a ruleset. This demonstrates how to differentiate the variables when there are multiple variables with the same paths within the same parent in the ruleset.

Sample Ruleset in Calculation Manager

Name	Application	Type	Description
Revenue Plan	Vision		
Copy Channel	Vision	Plan1	
Revenue Plan	Vision	Plan1	
Revenue Plan	Vision	Plan1	

Rule/RuleSet Name	Name	Scope	Is Hidden	Value	Use As Override Value
Copy Channel	Department	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
Revenue Plan	Department	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
Revenue Plan	Department	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>
	Version	Application	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>

Sample POST request

```
{
  "jobType": "Ruleset",
  "jobName": "Revenue Plan",
  "parameters": {
    "(1)Copy Channel.Department": "No Entity",
    "(1)Copy Channel.Version": "Worst Case",

    "(2)Revenue Plan.Department": "New Entity",
    "(2)Revenue Plan.Version": "Best Case",

    "(3)Revenue Plan.Version": "What If",
    "(3)Revenue Plan.Department": "Unspecified Entity"
  }
}
```

Plan Type Map

Copies data from a block storage database to an aggregate storage database or from a block storage to another block storage based on the settings specified in a Planning job of type plan type map.

See for details about plan type maps.

This API is not supported for Financial Consolidation and Close or Tax Reporting.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-12 Plan Type Map

Name	Description	Required	Default
<code>jobType</code>	Plan Type Map or <code>PLAN_TYPE_MAP</code> (both parameters are supported)	Yes	None
<code>jobName</code>	The name of a job of type plan type map that is already defined in the Planning application. Example: <code>CampaignToReporting</code>	Yes	None
<code>parameters</code>	Optionally, you can specify the <code>clearData</code> parameter that indicates whether the data in the target database should be moved before copying data. If this parameter value is not set, the default value <code>True</code> is used. The value must use JSON syntax. Example: <code>{clearData:false}</code>	No	True

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/v3/applications/PS4appl/jobs
```

```
{"jobType":"PLAN_TYPE_MAP","jobName":"MapReporting","parameters":  
{"cubeLinkName":"name","clearData":true}}
```

Import Data

Imports data from a file in the Planning repository into the application using the import data settings specified in a Planning job of type `Import Data`.

You can also override some of the parameters of the job definition while executing this job from a REST API.

For Planning, Financial Consolidation and Close, and Tax Reporting, you can review the rejected data records that have errors. To do this, specify an error file that captures the data records that are not imported for each dimension. If an error file is specified, the ZIP file is stored in the Outbox where you can download the file using Inbox/Outbox Explorer or tools like EPM Automate or REST APIs, for example, with the [Download API](#).

Required Roles

Service Administrator

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-13 Import Data

Name	Description	Required	Default
<code>jobType</code>	Import Data or <code>IMPORT_DATA</code> (both parameters are supported)	Yes	None
<code>jobName</code>	The name of a job of type import data exactly as it is already defined in the Planning application. Example: <code>dailydataload</code>	Yes	None
<code>importFileName</code>	Optionally, you can specify the name of the ZIP, CSV or TXT (Essbase format data file) file from which data is to be imported. If you specify a file name, the import file name in the job is ignored. If the job is defined to import data in Essbase format, the ZIP file must contain an Essbase format TXT file. For other import jobs, the ZIP file may contain one or more CSV files that identify the import sequence in the file names; for example <code>data1-3.csv</code> , <code>data2-3.csv</code> , and <code>data3-3.csv</code> . The value must use JSON syntax.	No	The source file of the job definition.
<code>sourceType</code>	You can override the source type to be used. Allowed value is <code>Planning</code> or <code>Essbase</code> .	No	The Source Type parameter of the job definition.
<code>delimiter</code>	You can override the delimiter to be used. Allowed value is <code>comma</code> or <code>tab</code> . This is only applicable when the source type is <code>Planning</code> .	No	The delimiter used in the job definition.
<code>dateFormat</code>	You can override the date format to be used. This is applicable only when the source type is <code>Planning</code> . Allowed value is one of the following: <code>MM-DD-YYYY</code> , <code>DD-MM-YYYY</code> , or <code>YYYY-MM-DD</code> .	No	The Date Format parameter of the job definition.
<code>includeMetaData</code>	You can override the option to include metadata. Allowed value is <code>true</code> or <code>false</code> . This is only applicable when the source type is <code>Planning</code> .	No	Include Metadata parameter of the job definition.
<code>cube</code>	You can override the name of the cube defined in the job definition. This is only applicable when the source type is <code>Essbase</code> .	No	Cube parameter of the job definition.

Table 7-13 (Cont.) Import Data

Name	Description	Required	Default
errorFile	Optionally, create an error file. The error file is zipped with the name of this parameter. The ZIP file is stored in the Outbox. You can download it with the Download API . This API overrides any existing error file with the same name. Example: ImportDataErrorFile.zip	No	No error files are created
stopOnError	Optionally, use the <code>stopOnError</code> parameter to stop the import process if an intermediate error is encountered during the import, for example, if data load finds an unknown member or invalid data value. This setting can only be used when <code>sourceType</code> is <code>Essbase</code> . When using this option, if you specified an error file, you can then review the rejected data record that has the error. The record is included in a ZIP file that is stored in the outbox where you can download it for review.	No	true

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payloads

Example 1: Executes the import data job `ImportJob` and overrides the `importFileName` parameter.

```
{ "jobType": "IMPORT_DATA", "jobName": "ImportJob",
  "parameters": {
    "importFileName": "myImportfile123.zip"
  }
}
```

Example 2: Executes the import data job `ImportJob` and overrides the `delimiter`, `dateFormat`, and `includeMetaData` parameters.

```
{ "jobType": "IMPORT_DATA", "jobName": "ImportJob",
  "parameters": {
    "delimiter": "comma",
    "dateFormat": "MM-DD-YYYY",
    "includeMetaData": "false"
  }
}
```

Example 3: Executes the import data job `ImportJob` defined with `sourceType` as `Essbase` and overrides the `sourceType` and `cube` parameters.

```
{ "jobType": "IMPORT_DATA", "jobName": "ImportJob",
  "parameters": {
    "sourceType": "Essbase",
    "cube": "Plan1"
  }
}
```

Example 4: Executes the ImportData job ImportDataJob and overrides the `errorFile` parameter with a value `ImportDataErrorFile.zip`. If error records are found during the Import Data operation, a ZIP file called `ImportDataErrorFile.zip` is created in the Planning repository. The generated error file can be downloaded from the Outbox from the job status page or using the Download REST API or EPM Automate `downloadfile` command.

```
{
  "jobType": "IMPORT_DATA",
  "jobName": "ImportDataJob",
  "parameters": {
    "errorFile": "ImportDataErrorFile.zip"
  }
}
```

Example 5: Executes the ImportData job `ImportDataJob_Sample` defined with `sourceType` as `Essbase`, and overrides the `stopOnError` parameter with the value as `true`. The data load will stop loading in case of an intermediate error.

```
{
  "jobType": "IMPORT_DATA",
  "jobName": "ImportDataJob_Sample",
  "parameters": {
    "importFileName": "importDataFile_Essabse.txt",
    "cube": "Plan1",
    "stopOnError": "true"
  }
}
```

Export Data

Exports application data into a file using the export data settings, including file name, specified in a Planning job of type export data. The file containing the exported data is stored in the Planning repository.

You can also override some of the parameters of the job definition while executing this job with a REST API.

Exporting data supports substitution variables. You can use substitution variables while overriding the `rowMembers`, `columnMembers`, and `povMembers` definition. See *Exporting Data and Creating and Assigning Values to Substitution Variables in Administering Planning*.

Required Roles

Service Administrator

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-14 Export Data

Name	Description	Required	Default
jobType	Export Data or EXPORT_DATA (both parameters are supported)	Yes	None
jobName	The name of a job of type export data exactly as it is already defined in the Planning application. Example: dailydataexport	Yes	None
exportFileName	You can specify a file name for the exported data. Data is exported as a ZIP file only. The exported file is stored in the Planning repository. The value must use JSON syntax.	No	The file name for the exported data will be the same as Job Name.
delimiter	You can use this parameter to override the delimiter parameter. Allowed value is comma or tab.	No	Delimiter used in the job definition.
exportSmartListAs	You can use this parameter to override how Smart Lists are exported. Allowed value is label or name.	No	Smart Lists parameter of the job definition.
includeDynamicMembers	You can use this parameter to include or exclude dynamic members. Allowed value is true or false.	No	Dynamic Members parameter of the job definition.
cube	You can use this parameter to run the export data job on a different cube.	No	Cube parameter of the job definition.
rowMembers	You can override the row members defined in the job definition. Example: "Name,Price,Discount %"	No	Row parameter defined in the Slice section of the job definition.
columnMembers	You can override the column members defined in the job definition. Example: "Name,Price,Discount %"	No	Column parameter defined in the Slice section of the job definition.
povMembers	You can override the POV members defined in the job definition. Example: "Name,Price,Discount %"	No	Point of View parameter defined in the Slice section of the job definition.
exportDataDecimalScale	Specify the number of decimal positions (0-16) that will be returned when exporting data from Essbase. If the default None is selected, the data that is returned will not be formatted and will return as Essbase returns it. Selecting a numeric value will result in the exported data displaying that number of digits to the right of the decimal point, wherever applicable. For example, specifying 3 in the Decimals field will result in the exported data being formatted to display three digits to the right of the decimal point. Selecting 0 formats the data to display a whole number. Example: A value 27.07000001 can be formatted and written as 27.07 in the export data file if the value for this parameter is set to 2.	No	Decimals parameter of the job definition

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payloads

Example 1: Executes the export data job `ExportJobDaily` and overrides the `exportFileName` parameter.

```
{"jobType":"EXPORT_DATA","jobName":"ExportJobDaily",
  "parameters":{
    "exportFileName":"myExportfile.zip"
  }
}
```

Example 2: Executes the export data job `ExportJobDaily` and overrides the `delimiter`, `exportSmartListAs`, and `includeDynamicMembers` parameters.

```
{"jobType":"EXPORT_DATA","jobName":"ExportJobDaily",
"parameters":{
  "delimiter":"tab",
  "exportSmartListAs":"name",
  "includeDynamicMembers":"true"
}
```

Example 3: Executes the export data job `ExportJobDaily` and overrides the `cube` parameter only. This job will now execute for the cube `Vis1ASO`.

```
{"jobType":"EXPORT_DATA","jobName":"ExportJobDaily",
"parameters":{
  "cube":"Vis1ASO"
}
```

Example 4: Executes the export data job `ExportJobDaily` and overrides the `cube` name along with the `rowMembers`, `columnMembers`, and `povMembers` parameters. This job will now execute for the cube `Vis1ASO`.

```
{"jobType":"EXPORT_DATA","jobName":"ExportJobDaily",
"parameters":{
  "cube":"Vis1ASO",
  "rowMembers":"Current,Variance,Actual,Scenario",
  "columnMembers":"Statistics,Account",
  "povMembers":"Period,Year,Version,Entity,Product,Channel"
}
```

Example 5: Executes the export data job `ExportJobDaily` and overrides the `cube` name along with the parameters `rowMembers`, `columnMembers`, and `povMembers`. We use substitution variables while overriding the `rowMembers`, `columnMembers`, and `povMembers` definition. This job executes for the cube `Vis1ASO`.

```
{"jobType":"EXPORT_DATA","jobName":"ExportJobDaily",
  "parameters":{
```

```

    "cube": "Vis1ASO",
    "rowMembers": "ILvl0Descendants (&Param1) ",
    "columnMembers": "ILvl0Descendants (&Param2) ",
    "povMembers": "Period, Year, Version, &Param3, Product, Channel"
  }
}

```

Example 6: Executes the export data job `ExportJobDaily` and overrides the cube name along with the parameters `exportDataDecimalScale`, `rowMembers`, and `columnMembers`. We use substitution variables while overriding the `rowMembers`, `columnMembers`, and `povMembers` definition. This job executes for the cube `Vis1ASO`.

```

{"jobType": "EXPORT_DATA", "jobName": "ExportJobDaily",
 "parameters": {
   "cube": "Vis1ASO",
   "rowMembers": "ILvl0Descendants (&Param1) ",
   "columnMembers": "ILvl0Descendants (&Param2) ",
   "povMembers": "Period, Year, Version, &Param3, Product, Channel",
   "exportDataDecimalScale": "2"
 }
}

```

Import Metadata

Imports metadata from a file in the Planning repository into the application using the import metadata settings specified in a Planning job of type `import metadata`.

You can also override some of the parameters of the job definition while executing this job from a REST API.

For Planning, Financial Consolidation and Close, and Tax Reporting, you can specify an error file that captures the metadata records that are not imported for each dimension. If an error file is specified, a separate error file is created for each dimension. The error files are then zipped together and the zip file is stored in the Outbox where you can download the file using Inbox/Outbox Explorer or tools like EPM Automate or REST APIs, for example, with the [Download API](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-15 Import Metadata

Name	Description	Required	Default
jobType	Import Metadata or IMPORT_METADATA (both parameters are supported)	Yes	None
jobName	The name of a job of type import metadata exactly as it is already defined in the Planning application. Example: importAccount	Yes	None
importZipFileName	Optionally, you can specify the name of the ZIP file from which metadata is to be imported. The contents of the ZIP file that you specify take precedence over the file names defined in the job. The ZIP file may contain one or more CSV files. The file names containing metadata for dimensions should match the import file names defined in the job or end with <code>_DIMENSIONNAME.csv</code> ; for example, <code>metadata_Entity.csv</code> , <code>metadata_HSP_SmartLists.csv</code> , and <code>metadata_Exchange Rates.csv</code> . Only metadata for the dimensions for which metadata import is set up in the job is imported. Metadata for other dimensions, if contained in the ZIP file, is ignored. Example: {importZipFileName:importAccount.zip}	No	The import file name defined in the job definition.
refreshCube	You can override the option to perform a refresh cube action defined in the job. Allowed values are either <code>true</code> or <code>false</code> .	No	"Refresh Database if Import Metadata is successful" parameter of the job definition.
errorFile	Optionally, create a separate error file for each dimension. The error files are zipped with the name of this parameter. The ZIP file is stored in the Outbox. You can download it with the Download API. This API overrides any existing error file with the same name Example: ImportMetaDataErrorFile.zip	No	No error files are created.

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example: Executes the job `ImportMetaDataJob` and overrides only the `importZipFileName` parameter.

```
{
  "jobType": "IMPORT_METADATA",
  "jobName": "ImportMetaDataJob",
  "parameters": {
    "importZipFileName": "myMetaDataDailyJob.zip"
  }
}
```

Example: Executes the job `ImportMetaDataJob` and overrides the `errorFile` parameter with a value `ImportMetaDataErrorFile.zip`. If there are error records found during the Import Metadata operation for one or more dimensions, a ZIP file called `ImportMetaDataErrorFile.zip` is created in the repository that contains one error CSV file for each failed dimension. The generated error file can be downloaded from the Outbox from the job status page or using the [Download](#) REST API or EPM Automate `downloadfile` command.

```
{
  "jobType": "IMPORT_METADATA",
  "jobName": "ImportMetaDataJob",
  "parameters": {
    "errorFile": "ImportMetaDataErrorFile.zip"
  }
}
```

Export Metadata

Exports metadata into a file using the settings specified in a Planning job of type `export metadata`. The file containing the exported metadata is stored in the Planning repository.

You can also override some of the parameters of the job definition while executing this job from a REST API.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-16 Export Metadata

Name	Description	Required	Default
<code>jobType</code>	<code>Export Metadata</code> or <code>EXPORT_METADATA</code> (both parameters are supported)	Yes	None
<code>jobName</code>	The name of a job of type export metadata exactly as it is already defined in the Planning application. Example: <code>dailyAccountExport</code>	Yes	None
<code>exportZipFileName</code>	Optionally, you can specify a file name for the exported metadata. Metadata is exported as a ZIP file only. The value must use JSON syntax. Example: { <code>exportZipFileName:Accountexport.zip</code> }	No	The file name for the exported metadata will be the same as Job Name

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Example Payload

Example: Executes the export metadata job "ExportMetadataDaily" and overrides the `exportZipFileName` parameter.

```
{
  "jobType": "EXPORT_METADATA",
  "jobName": "ExportMetadataDaily",
  "parameters": {
    "exportZipFileName": "dailyMetaData.zip"
  }
}
```

Cube Refresh

Refreshes the Planning application cube. Typically, you refresh the cube after importing metadata into the application.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-17 Cube Refresh

Name	Description	Required	Default
<code>jobType</code>	Cube Refresh or CUBE_REFRESH (both parameters are supported)	Yes	None
<code>jobName</code>	Name of the job to run. You must use the exact name of a job that is already defined in the application. Example: refreshcube	Yes	None
<code>allowedUsersDurationCubeRefresh</code>	Possible values: Administrators or All Users	No	None

Table 7-17 (Cont.) Cube Refresh

Name	Description	Required	Default
terminateActiveRequestsBeforeCubeRefresh	Possible values: true or false	No	None
logOffAllUsersBeforeCubeRefresh	Possible values: true or false	No	None
allowedUsersAfterCubeRefresh	Possible values: Administrators or All Users	No	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

```
{"jobType":"CUBE_REFRESH","jobName":"CubeRefresh"}
```

Sample Payload overriding parameters:

```
{"jobType":"CUBE_REFRESH","jobName":"MyRefreshCube","parameters":{"allowedUsersDuringCubeRefresh":" All Users","terminateActiveRequestsBeforeCubeRefresh":"false","logOffAllUsersBeforeCubeRefresh":"true","allowedUsersAfterCubeRefresh":"Administrators"}}
```

Clear Cube

Enables you to clear specific data within input and reporting cubes.

You can clear the data using member selection or a valid MDX query. Using member selection, you can also optionally clear related supporting details, comments, and attachments. You can also elect to do a physical or logical clear of data. This gives you more flexibility and granularity when clearing the cube.

NOTE: The Clear Cube job deletes the data you specify within input and reporting cubes, but it does not delete the application definition in the application's relational tables.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-18 Clear Cube

Name	Description	Required	Default
jobType	Clear Cube or CLEAR_CUBE (both parameters are supported)	Yes	None
jobName	Name of the job to run. Important: You must use the exact name of a job that is already defined in the application. Example: ClearPlan1	Yes	None
cube	Valid ASO cube name	No	As per the job definition
members	Valid member selection that is comma separated. Applicable only for Partial Clear Job, for an ASO cube, defined with member selection. Example: "ILvl0Descendants (Exchange Rates), Total Expense"	No	As per the job definition
mdxQuery	Valid MDX query. Applicable only for Partial Clear Job, for an ASO cube, defined with MDX query support. Example: "Crossjoin({ [Apr], [May], [Jun] }, { [Expense1] })"	No	As per the job definition
clearSupportingDetails	Specify if supporting details should be cleared. Allowed values: true or false. Applicable only for Partial Clear Job, for an ASO cube, defined with member selection.	No	As per the job definition

Table 7-18 (Cont.) Clear Cube

Name	Description	Required	Default
clearComments	Specify if comments should be cleared. Allowed values: true or false. Applicable only for Partial Clear Job, for an ASO cube, defined with member selection.	No	As per the job definition
clearAttachments	Specify if attachments should be cleared. Allowed values: true or false. Applicable only for Partial Clear Job, for an ASO cube, defined with member selection.	No	As per the job definition
clearPhysicalOnEssbase	Specify if this is a physical clear on Essbase. Allowed values: true or false. Applicable only for Partial Clear Job, for an ASO cube, defined with member selection or MDX query support.	No	As per the job definition

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

```
{"jobType":"CLEAR_CUBE", "jobName":"ClearPlan1"}
```

Example: Executes the clear job `Clear_Partial_Basic`, which is defined with member selection, and overrides the `cube` and `members` parameters.

```
{
  "jobType": "Clear Cube",
  "jobName": "Clear_Partial_Basic",
  "parameters": {
    "cube": "HP1_ASO",
    "members": "ILvl10Descendants(Exchange Rates),Siblings(Total Expense)"
  }
}
```

Example: Executes the clear job `Clear_Partial_Basic`, which is defined with member selection, and overrides the `cube`, `clearSupportingDetails`, `clearComments`, and other parameters.

```
{
  "jobType": "Clear Cube",
  "jobName": "Clear_Partial_Basic",
```

```

    "parameters": {
      "cube": "HP1_ASO",
      "members": "ILvl10Descendants(Exchange Rates), Siblings(Total
Expense)",
      "clearSupportingDetails": "false",
      "clearComments": false,
      "clearAttachments": false,
      "clearPhysicalOnEssbase": false
    }
  }
}

```

Example: Executes the clear job `Clear_Partial_Advanced`, which is defined with the MDX query, and overrides the `mdxQuery` parameter.

```

{
  "jobType": "Clear Cube",
  "jobName": "Clear_Partial_Advanced",
  "parameters": {
    "mdxQuery": "Crossjoin({[Apr], [May], [Jun]}, {[Expense1]})"
  }
}

```

Example: Executes the clear job `Clear_Partial_Advanced`, which is defined with the MDX query, and overrides the `cube`, `mdxQuery`, and `clearPhysicalOnEssbase` parameters.

```

{
  "jobType": "Clear Cube",
  "jobName": "Clear_Partial_Advanced",
  "parameters": {
    "cube": "HP1_ASO",
    "mdxQuery": "Crossjoin({[Apr], [May], [Jun]}, {[Expense1]})",
    "clearPhysicalOnEssbase": false
  }
}

```

Administration Mode

Changes the login level for a Planning application. If you set login level to Administrators, all Interactive Users and Planners will be logged off of the application upon completion of the job. For details on administration mode, see [Scheduling Jobs](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-19 Parameters

Name	Description	Required	Default
jobType	Administration Mode	Yes	None
jobName	The job name to be used for this job execution. Example: AppAdminJob	No	Administration Mode
loginLevel	Specify the login level for users using loginLevel. Possible values are Administrators or All Users	Yes	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example: This request will change the login level of the application to "Administrators" level.

```
{
  "jobType": "Administration Mode",
  "jobName": "AppAdminJob",
  "parameters": {
    "loginLevel": "Administrators"
  }
}
```

Compact Cube

Compacts the outline file of an ASO cube. Compaction helps keep the outline file at an optimal size. Compacting the outline does not clear the data. For more information, see [Scheduling Jobs](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-20 Parameters

Name	Description	Required	Default
jobType	Compact Cube	Yes	None
jobName	The job name to be used for this job execution. Example: CompactCube	No	Compact Outline
cubeName	Name of the ASO cube	Yes	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example: This request will compact the outline of Vis1ASO ASO cube.

```
{
  "jobType": "Compact Cube",
  "jobName": "CompactCube",
  "parameters": {
    "cubeName": "Vis1Aso"
  }
}
```

Restructure Cube

Performs a full restructure of a BSO cube to eliminate or reduce fragmentation. For more information, see [Scheduling Jobs](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-21 Parameters

Name	Description	Required	Default
jobType	Restructure Cube	Yes	None
jobName	The job name to be used for this job execution. Example: RestructureCube	No	Restructure Cube
cubeName	Name of the BSO cube	Yes	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example: This request will restructure Plan1 BSO

```
{
  "jobType": "Restructure Cube",
  "jobName": "RestructureCube",
  "parameters": {
    "cubeName": "Plan1"
  }
}
```

Merge Data Slices

Merges incremental data slices of an ASO cube. Fewer slices improve a cube's performance. You can merge all incremental data slices into the main database slice or merge all incremental data slices into a single data slice without changing the main database slice. You can optionally remove cells that have a value of zero. For more information, see [Scheduling Jobs](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-22 Parameters

Name	Description	Required	Default
jobType	Merge Data Slices	Yes	None

Table 7-22 (Cont.) Parameters

Name	Description	Required	Default
jobName	The job name to be used for this job execution. Example: MergeDataSlice	No	Merge Data Slices
cubeName	Name of the ASO cube	Yes	None
keepZeroCells	Possible values are true or false	Yes	None
mergeSliceType	Possible values are allIncrementalSlicesInMain (Merge all into the main slice) Or allIncrementalSlicesInOneIncremental (Merge all incremental into a single incremental slice)	Yes	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example: This request will merge all incremental data slices in the main data slice and keep zero value cells.

```
{
  "jobType": "Merge Data Slices",
  "jobName": "MergeDataSlice",
  "parameters": {
    "cubeName": "VisASO",
    "mergeSliceType": "allIncrementalSlicesInMain",
    "keepZeroCells": "true"
  }
}
```

Optimize Aggregation

Improves the performance of ASO cubes. This job has two actions: Enable query tracking and Execute aggregation process. It performs an aggregation, optionally specifying the maximum disk space for the resulting files, and optionally basing the view selection on user querying patterns. This API is only applicable to aggregate storage databases. For information about scheduling jobs, see [Scheduling Jobs](#).

Before using this API, you must first enable query tracking to capture tracking statistics on the aggregate storage cube. Then, after you enable query tracking, you must allow sufficient time to collect user data-retrieval patterns before you execute the aggregation process based on query data. The execute aggregation process deletes existing aggregated views and generates optimized views based on the collected query tracking data.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
POST /HyperionPlanning/rest/{api_version}/applications/{application}/
jobs/{jobId}
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-23 Parameters

Name	Description	Required	Default
jobType	Optimize Aggregation	Yes	None
jobName	The job name to be used for this job execution	No	Optimize Aggregation
parameters	Parameters required for the job	Yes	None
cubeName	Name of the ASO cube	Yes	None
type	Can take one of these values: enableQueryTracking or executeAggregationProcess	Yes	None
useQueryData	Aggregate the views the server selects based on collected user querying patterns. This option is only available if query tracking is turned on. Permissible values: true or false . Default: false . Applicable only if type is executeAggregationProcess .	No	None

Table 7-23 (Cont.) Parameters

Name	Description	Required	Default
includeAlternateRollups	Include secondary hierarchies (with default level usage) in the view selection. Permissible values: disable or enable . Default: disable (no secondary hierarchies are considered). Applicable only if type is executeAggregationProcess .	No	None
growthSizeRatio	Aggregates the views the server selects until the maximum growth of the aggregated database exceeds the limits you specify. The value can be a real number such as 1.01. Default is that the database will grow without any growth ratio limit. Applicable only if type is executeAggregationProcess .	No	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: This request will enable query tracking on the Vis1ASO cube.

```
{
  "jobType": "Optimize Aggregation",
  "jobName": "CubeOptimizeAggr",
  "parameters": {
    "cubeName": "Vis1ASO",
    "type": "enableQueryTracking"
  }
}
```

Example 2: This request will execute the aggregation process on the Vis1ASO cube.

```
{
  "jobType": "Optimize Aggregation",
  "jobName": "CubeOptimizeAggr",
  "parameters": {
    "cubeName": "Vis1ASO",
    "type": "executeAggregationProcess"
  }
}
```

```
    }
  }
```

Example 3: This request will execute aggregation process on the Vis1ASO cube. Aggregation process will use the query tracking data, will not include alternate roll ups, and use growth size ratio as 1.01.

```
{
  "jobType": "Optimize Aggregation",
  "jobName": "CubeOptimizeAggr",
  "parameters": {
    "cubeName": "Vis1ASO",
    "type": "executeAggregationProcess",
    "useQueryData": "true",
    "includeAlternateRollups": "disable",
    "growthSizeRatio": "1.01"
  }
}
```

Import Security

Imports the security records or access control list (ACL) records from a Comma Separated Values (CSV) file. For information about access permissions to application artifacts, see [Setting Up Access Permissions](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Excel File Format

- Object Name: The name of the object for which the ACL is defined
- Name: The name of the user or group for which the ACL is defined
- Parent: The name of the parent of the object
- Is User: The flag (Y or N) that determines if the ACL is defined for a user or for a group
- Object Type: The type of object, for example, `Forms folder`
- Access Type: The type of privilege, such as READ or READWRITE
- Access Mode: Additional information, such as if the ACL is applicable on the descendants
- Remove: To remove a particular ACL, set this to Y

All possible values:

Object Type:

- SL_FORM - Form
- SL_COMPOSITE - Composite Form
- SL_TASKLIST - Tasklist
- SL_CALCRULE - Rule
- SL_FORMFOLDER - Form Folder
- SL_CALCOLDER - Rule Folder

- SL_DIMENSION - Dimension
- SL_CALCTEMPLATE - Template
- SL_REPORT - Management Report
- SL_REPORTSHOT - Management Report Snapshot

Access Type:

- NONE - None
- READ - Read
- WRITE - Write
- READWRITE - Read Write
- LAUNCH - Launch

Access Mode:

- MEMBER
- CHILDREN
- @ICHILDREN
- @DESCENDANTS
- @IDESCENDANTS

CSV File Example:

```
Object Name,Name,Parent,Is User,Object Type,Access Type,Access Mode,Remove
"Exchange Rates to USD","ats_user3","Y","SL_FORM","READWRITE","MEMBER","N"
"Exchange Rates to USD","ats_user4","Y","SL_FORM","READWRITE","MEMBER","N"
"Exchange Rates to USD","ats_user15","Y","SL_FORM","READ","MEMBER","N"
"Exchange Rates to USD","ats_user10","Y","SL_FORM","NONE","MEMBER","N"
"Calculate Benefits","group_1","N","SL_COMPOSITE","READWRITE","MEMBER","N"
```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-24 Parameters

Name	Description	Required	Default
jobType	Import Security	Yes	None

Table 7-24 (Cont.) Parameters

Name	Description	Required	Default
jobName	The job name to be used for this job execution. Example: ImportSecurity	No	Import Security
fileName	The input CSV file for import. The file containing the ACL records should be present in the Planning Cloud repository.	Yes	None
clearAll	Clear existing access permissions when importing new access permissions. Possible values are true or false	No	False
errorFile	Optionally, create a separate error file for recording any errors that occur during the import process. The file containing the error messages is stored in the Outbox. You can download it with the Download API . This API overrides any existing error file with the same name.	No	None

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Imports security records from the input file `ImportSecurityRecordsFile.csv`. Existing security records are retained.

```
{
  "jobType": "Import Security",
  "jobName": "ImportSecurity",
  "parameters": {
    "fileName": "ImportSecurityRecordsFile.csv"
  }
}
```

Example 2: Imports security records from the input file

ImportSecurityRecordsFile.csv. Clears existing security records before importing.

```
{
  "jobType": "Import Security",
  "jobName": "ImportSecurity",
  "parameters": {
    "fileName": "ImportSecurityRecordsFile.csv"
    "clearAll": "true"
  }
}
```

Example 3: Imports security records from the input file

ImportSecurityRecordsFile.csv and exports the error messages to the file SecurityImportErrors.txt. Existing security records are retained.

```
{
  "jobType": "Import Security",
  "jobName": "ImportSecurity",
  "parameters": {
    "fileName": "ImportSecurityRecordsFile.csv"
    "clearAll": "true"
    "errorFile": "SecurityImportErrors.txt"
  }
}
```

Export Security

Exports the security records or access control list (ACL) records for specified users or groups to a Comma Separated Values (CSV) file. For information about access permissions to application artifacts, see [Setting Up Access Permissions](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-25 Parameters

Name	Description	Required	Default
jobType	Export Security	Yes	None

Table 7-25 (Cont.) Parameters

Name	Description	Required	Default
jobName	The name of this job. Example: ExportSecurity	No	Export Security
fileName	The name of the file to which records should be exported. The file containing the exported data is stored in the Planning repository. If fileName is not specified, a file is auto-generated with a name containing the user name, current date, and time stamp. For example, test_admin_SecurityRecords_2019-02-26-04-34-45-420.csv.	No	The file name is auto-generated
exportUsers	Comma separated user names. Only ACL records related to the specified users are exported.	No	None
exportGroups	Comma separated user names. Only ACL records related to the specified groups are exported.	No	None

Notes:

- This job can take only `exportGroups` or `exportUsers` at one time. If you need to export groups and users, you must run the job twice, once with each parameter.
- If a user name or group name contains a comma, escape the comma in the request. For example, if a user name is `test,User`, the request should contain `test\\,User`.
- For the file format, see the definition in [Import Security Records](#).

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Exports all security records to the `ExportSecurityRecordsFile.csv` file.

```
{
  "jobType": "Export Security",
  "jobName": "ExportSecurity",
  "parameters": {
    "fileName": "ExportSecurityRecordsFile.csv"
  }
}
```



```
}  
}
```

Example 2: Exports security records of two groups, group1 and group2, to the ExportSecurityRecordsFile.csv file.

```
{  
  "jobType": "Export Security",  
  "jobName": "ExportSecurity",  
  "parameters": {  
    "exportGroups": "group1,group2"  
    "fileName": "ExportSecurityRecordsFile.csv"  
  }  
}
```

Example 3: Exports security records of two users, test1 and test,User2 to the ExportSecurityRecordsFile.csv file. Note that one user name contains a comma in it.

```
{  
  "jobType": "Export Security",  
  "jobName": "ExportSecurity",  
  "parameters": {  
    "exportUsers": "test1,test\\,User2"  
    "fileName": "ExportSecurityRecordsFile.csv"  
  }  
}
```

Export Audit

Exports the audit records to a Comma Separated Values (CSV) file. The output CSV file contains the first character as a Byte Order Mark (BOM) character `\u00ff`. The API writes an encrypted application identifier following the BOM character. This application identifier is written between double-quotes. Headers for the CSV file follow the application identifier. For more information, see [Auditing Tasks and Data](#).

You can use an optional `excludeApplicationId` parameter to not write the application identifier in the export file. Exported audit reports without the application identifier cannot be imported back into the application.

The generated CSV file is compressed and the output is a ZIP file. The file can be downloaded using the [Download REST API](#).

Required Roles

Service Administrator

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-26 Parameters

Name	Description	Required	Default
jobType	Export Audit	Yes	None
jobName	The job name to be used for this job execution. Example: ExportAudit	No	Export Audit
userNames	Comma separated user names. Audit records created by only the specified users are exported. If not specified, audit records created by all users are exported.	No	None
fileName	The name of the file to which records should be exported. The file containing the exported data is stored in the Oracle Planning and Budgeting Cloud repository. If fileName is not specified, a file is auto-generated with a name containing the user name, current date, and time stamp. For example, test_admin_AuditRecords_2019-02-26-04-34-45-420.zip	No	The file name is auto-generated

Table 7-26 (Cont.) Parameters

Name	Description	Required	Default
nDays	<p>Number of days for which audit records should be exported. Possible values are:</p> <ul style="list-style-type: none"> • 1: Export audit records for the last 24 hours • 2: Export audit records for the last two days • 7: Export audit records for the last seven days • 30: Export audit records for the last 30 days • 60: Export audit records for the last 60 days • 180: Export all existing audit records for the last 180 days • All: Export all audit records <p>Note: Audit information is maintained for up to 365 days from the current system date.</p>	No	7
excludeApplicationIdentifier	<p>Optionally, you can specify if the application identifier should be written in the export file. Possible values: true or false</p> <p>This parameter can benefit those customers who do not want the application identifier to be included in the export file to help import into their own systems. This export file cannot be used to import into an EPM Cloud environment.</p>	No	false

Notes:

- This job does not export records based on group names.
- If a user name contains a comma, escape the comma in the request. For example, if a user name is `test,User` then add `test\\,User` to the request.

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Exports the last 180 days of audit records to the `ExportAuditLast180Days.zip` file.

```
{
  "jobType": "Export Audit",
  "jobName": "Export180DaysAudit",
  "parameters": {
    "fileName": "ExportAuditLast180Days.zip",
    "ndays": "180"
  }
}
```

Example 2: Exports the last seven days of audit records created by `planner1` and `planner2`. Records are exported to a zip file with an auto-generated file name.

```
{
  "jobType": "Export Audit",
  "jobName": "AllAuditRecordsOfPlanners",
  "parameters": {
    "userNames": "planner1, planner2"
  }
}
```

Example 3: Exports the last 180 days of audit records to the `ExportAuditLast180Days.zip` file. The application identifier will be not written in the generated file. This export file cannot be used to import into an EPM Cloud environment.

```
{
  "jobType": "Export Audit",
  "jobName": "Export180DaysAudit",
  "parameters": {
    "fileName": "ExportAuditLast180Days.zip",
    "ndays": "180",
    "excludeApplicationId": "true"
  }
}
```

Export Job Console

Exports the job console records to a Comma Separated Values (CSV) file. The output CSV file contains the first character as a Byte Order Mark (BOM) character, `\ufeff`. The API writes an encrypted application identifier following the BOM character. This application identifier is written between double-quotes. Headers for the CSV file follow the application identifier.

You can use an optional `excludeApplicationId` parameter to not write the application identifier in the export file. Exported job console data files without the application identifier cannot be imported back into the application.

The generated CSV file is compressed and the output is a ZIP file. The file can be downloaded using the Download REST API.

To view pending jobs, see [Viewing Pending Jobs and Recent Activity](#).

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-27 Parameters

Name	Description	Required	Default
<code>jobType</code>	Export Job Console	Yes	None
<code>jobName</code>	The job name to be used for this job execution. Example: <code>ExportWeeklyJobStatusRecords</code>	No	Export Job Console
<code>userNames</code>	Comma separated user names. Job console records created by only the specified users are exported. If not specified, job console records created by all users are exported.	No	None

Table 7-27 (Cont.) Parameters

Name	Description	Required	Default
fileName	The name of the file to which records should be exported. The file containing the exported data is stored in the Planning repository. If fileName is not specified, a file is auto-generated with a name containing the user name, current date, and time stamp. For example, test_admin_JobConsoleRecords_2019-02-26-04-34-45-420.zip	No	The file name is auto-generated
ndays	Number of days for which audit records should be exported. Possible values are. <ul style="list-style-type: none"> • 1: Export job console records for the last 24 hours • 2: Export job console records for the last two days • 7: Export job console records for the last seven days • 30: Export job console records for the last 30 days • 60: Export job console records for the last 60 days • All: Export all job console records 	No	7
jobNames	Comma-separated job names for which job console records should be exported.	No	None

Table 7-27 (Cont.) Parameters

Name	Description	Required	Default
jobTypes	<p>Comma-separated job codes for which job console records should be exported. Possible values are:</p> <ul style="list-style-type: none"> • ALL • Rules or RULES • Ruleset or RULESET • Clear Cell Details or CLEAR_CELL_DETAILS • Copy Data or COPY_DATA • Invalid Intersection Report/INVALID_INTERSECTION_RPT • Copy Versions or COPY_VERSIONS • Content Upgrade or CONTENT_UPGRADE • Plan Type Map or PLAN_TYPE_MAP • Import Data or IMPORT_DATA • Export Data or EXPORT_DATA • Export Metadata or EXPORT_METADATA • Import Metadata or IMPORT_METADATA • Cube Refresh or CUBE_REFRESH • Clear Cube or CLEAR_CUBE • Administration Mode or ADMIN_MODE • Compact Cube or COMPACT_CUBE • Restructure Cube or 	No	Rules

Table 7-27 (Cont.) Parameters

Name	Description	Required	Default
	RESTRUCTURE_CUBE <ul style="list-style-type: none"> • Merge Data Slices or MERGE_DATA_SLICES • Optimize Aggregation or OPTIMIZE_AGGREGATION • Import Security or SECURITY_IMPORT • Export Security or SECURITY_EXPORT • Export Audit or AUDIT_EXPORT • Export Job Console or JOB_CONSOLE_EXPORT • Sort Members or SORT_MEMBERS • Smart Push or SMART_PUSH • Import Exchange Rates or IMPORT_EXCHANGE_RATES • Execute Bursting Definition or EXECUTE_MR_BURST 		

Table 7-27 (Cont.) Parameters

Name	Description	Required	Default
jobStatusCodes	Comma-separated status code of the jobs for which job records are exported. Possible values are: 1 - Processing 2 - Completed successfully 3 - Failed with errors 4 - Completed with unknown status 5 - Completed with threshold violation status 6 - Pending cancellation 7 - Cancelled 8 - Completed with errors 9 - Completed with warnings all - All jobs with any status	No	2
exportErrorDetails	If true, exports the details of failed/error jobs as separate error log file in the final output file. Job status details of jobs having one of the status listed below are exported in separate files. <ul style="list-style-type: none"> • Error • Unknown • Completed with Threshold Exception • Completed with Errors • Completed with Warnings 	No	true

Table 7-27 (Cont.) Parameters

Name	Description	Required	Default
topCountDuration	Filters the top <i>n</i> job status records by completion time. This is useful for finding the longest running jobs. For example, enter 10 for this parameter to review the top 10 longest running jobs, as in this example: <code>"topCountDuration": "10"</code>	No	0 Represents all records
excludeApplicationId	Optionally, you can specify if the application identifier should be written in the export file. Possible values: true or false This parameter can benefit those customers who do not want the application identifier to be included in the export file to help import into their own systems. This export file cannot be used to import into an EPM Cloud environment.	No	false

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Exports the job console records for all default parameters into the `NewFile.csv` file. Exports status of all Rule jobs completed in the last seven days.

```
{
  "jobType": "JOBCONSOLE_EXPORT",
  "jobName": "AllJobConsoleExports",
  "parameters": {"fileName": "NewFile.csv"}
}
```

Example 2: Exports the job console records for Rule and Export Data jobs that completed normally or with an error status over the last month into the `exportFile.csv` file. Job details of the failed jobs are exported in separate files in the final compressed file.

```
{
  "jobType":"JOBCONSOLE_EXPORT",
  "parameters":{"fileName":"exportFile.csv", "jobTypes": "Rules,
EXPORT_DATA", "jobStatusCodes": "2,3", "ndays":"30"}
}
```

Example 3: Exports the job console records for the jobs named Daily Consolidation and Smart Push to a Reporting Cube. Includes jobs that completed normally or with an error status over the last month into the `exportFile.csv` file. Job details of the failed jobs are not exported in separate files because `exportErrorDetails` is `false`.

```
{
  "jobType":"JOBCONSOLE_EXPORT",
  "parameters":{"fileName":"exportFile.csv", "jobNames":"Daily
Consolidation, Smart Push to Reporting Cube", "jobStatusCodes": "2,3",
"ndays":"30", "exportErrorDetails":"false"}
}
```

Example 4: Exports the top 10 longest running jobs:

```
{
  "jobType":"JOBCONSOLE_EXPORT",
  "parameters":{"fileName":"exportFile.csv", "jobStatusCodes":
"all",
"ndays":"all", "jobTypes":"all", "exportErrorDetails":"false",
"topCountDuration":"10"}
}
```

Example 5: Exports the job console records for all default parameters into to the `NewFile.csv` file. This exports status of all Rule jobs completed in the last seven days. The application identifier will be not written in the generated file. This export file cannot be used to import into an EPM Cloud environment.

```
{
  "jobType":"JOBCONSOLE_EXPORT",
  "jobName":"AllJobConsoleExports",
  "parameters":{"
"fileName":"NewFile.csv",
"excludeApplicationId": "true"
}
}
```

Sort Members

Sorts the dimension members of a business process.

You can sort Entity, Account, Scenario, Version, and user-defined custom dimension members. You cannot sort Period, Years, or Currency dimension members. This

feature is only supported for the Planning, Module, and Free Form business processes. For more information, see [Sorting Members](#).

For Planning Module applications, you cannot sort these dimensions:

- Any dense dimension
- The "Plan Element" dimension, even if it is renamed, from Financials
- The "Project Element" dimension, even if it is renamed, from Projects

After sorting members, administrators must perform a cube refresh.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-28 Parameters

Name	Description	Required	Default
<code>jobType</code>	Sort Members	Yes	None
<code>jobName</code>	The job name to be used for this job execution. Example: SortEntity	No	Sort Members
<code>member</code>	Parent member whose children or descendants are being sorted.	Yes	None
<code>order</code>	Order of sorting. Possible values are: <ul style="list-style-type: none"> • ascending • descending 	No	ascending

Table 7-28 (Cont.) Parameters

Name	Description	Required	Default
type	Sort children or descendants. Possible values are: <ul style="list-style-type: none"> children - sorting by children affects only members in the level immediately below the specified member descendants - sorting by descendants affects all descendants of the specified member. 	No	children

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Sorts the child members of the Account dimension in ascending order.

```
{
  "jobType": "Sort Members",
  "jobName": "SortAccount",
  "parameters":
  {
    "member": "Account"
  }
}
```

Example 2: Sorts the child members of the Account dimension in descending order.

```
{
  "jobType": "Sort Members",
  "jobName": "SortAccountDesc",
  "parameters":
  {
    "member": "Account", "order": "descending"
  }
}
```

Example 3: Sorts the descendants of member account200 in descending order.

```
{
  "jobType": "Sort Members",
  "jobName": "SortAccount200Desc",
```

```

    "parameters":
    {
      "member": "account200", "order": "descending", "type": "descendants"
    }
  }
}

```

Import Exchange Rates

Export the Exchange Rates template in the Planning repository and change the rates if required. You can then import the rates into the application using the Import Exchange Rates settings specified in a Planning job of type import exchange rates. For more information, see [Job Types](#).

You can also override some of the parameters of the job definition while executing this job from a REST API.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-29 Import Exchange Rates

Name	Description	Required	Default
jobType	Import Exchange Rates or IMPORT_EXCHANGE_RATES (both parameters are supported)	Yes	None
jobName	The name of the job to be used for this job execution. Important: You must use the exact name of a job that is already defined in Planning as described in Managing Jobs . Example: importNewExchangeRate	Yes	None

Table 7-29 (Cont.) Import Exchange Rates

Name	Description	Required	Default
importFileName	Optionally, you can specify the name of the CSV file from which exchange rates are to be imported. If you specify a file name, the Import Exchange Rate file name in the job is ignored.	No	The source file of the job definition
includeMetaData	You can override the option to include metadata. Allowed value is <code>true</code> or <code>false</code> .	No	Include Metadata parameter of the job definition

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Executes the import exchange rates job `MyExchangeRates` and overrides the `importFileName` parameter.

```
{
  "jobType": "Import Exchange Rates",
  "jobName": "MyExchangeRates",
  "parameters": {
    "importFileName": "ExportExchangeRatesTemplate.csv"
  }
}
```

Example 2: Executes the import exchange rates job `MyExchangeRates` and overrides the `importFileName` and `includeMetaData` parameters.

```
{
  "jobType": "Import Exchange Rates",
  "jobName": "MyExchangeRates",
  "parameters": {
    "importFileName": "ExchangeRateTemplate2.csv",
    "includeMetaData": "false"
  }
}
```

Auto Predict

Schedule predictions using the Auto Predict job. With Auto Predict, administrators can define a prediction to predict future performance based on historical data and schedule a job to run that prediction definition, automating the prediction process. For details about Auto Predict, see [Setting Up Predictions to Run Automatically](#) in *Administering Planning*.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Job Types](#).

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 7-30 Import Exchange Rates

Name	Description	Required	Default
jobType	Auto Predict	Yes	None
jobName	The name used to define Auto Predict in the user interface in Overview, then Actions, and then Auto Predict. This name will be used for the job execution. Important: You must use the exact name of a job that is already defined in Planning as described in Managing Jobs . Example: Prediction1	Yes	None
forceRun	If this is set to true, the job will always execute. If not, the job executes only for the first time. In subsequent attempts, if there is no change in the job definition, this message displays: "Auto Prediction definition hasn't changed since the last time the Auto Predict job ran; the job will not execute. If you want to run the Auto Predict definition, for example if there are new actual values, you can run the Auto Predict definition from the Auto Predict page. From the Actions menu, click Actions and then click Run."	No	False

Table 7-30 (Cont.) Import Exchange Rates

Name	Description	Required	Default
paginatedDim	Speeds up an Auto Predict job by running predictions in parallel in separate prediction threads. For the parallel jobs to be efficient, choose a dimension that will result in evenly spread data for each prediction thread. Example: Entity	No	False

For a sample URL, see Sample URL and Payload in [Execute a Job](#).

Sample Payload

Example 1: Executes the Auto Predict job ASO->BSO.

```
{
  "jobType": "Auto Predict",
  "jobName": "ASO->BSO",
  "parameters": {
    "forceRun": true,
    "paginatedDim": "Entity"
  }
}
```

Import Cell-Level Security

This REST API imports cell-level security settings from a ZIP file that contains an Excel file with cell-level security definitions into Planning or Tax Reporting business processes. Cell-level security enables Service Administrators to restrict who can view data in the application by defining rules that remove read or write access to cells that a user would normally have access to due to their regular security.

The file must be present in the Inbox. You can use the Upload REST API to upload the file. Any rejected records are generated in an Excel file that is zipped and copied to the Outbox.

The best method to get the import file format template is to export cell-level security from the application.

The following is a general explanation of the Excel file. The file contains two Excel worksheets:

1. **Rules** - contains cell-level security definitions, dimensions included, properties such as Unspecified Valid, and Additional Dims Required
2. **Sub Rules** - contains member selections and exclusions

The **Rules** worksheet has the following column headings:

- **Name**
- **Position**
- **Description**
- **Enabled**
- **Valid Cubes** - This column can contain either `All` or a list of comma-separated names of cubes, such as `Plan1, Plan2`
- **Anchor Dim Name**
- **Anchor Dimension Apply to Unselected Members**
- **Dim1**
- **Dim1 Required**
- **Dim2**
- **Dim2 Required**
- **Dim X**
- **Dim X Required**

The `Sub Rules` worksheet has the following column headings:

- **Name** - This column must contain the name of the Rules from the first worksheet
- **Users**
- **User Groups**
- **Restriction** - This column can contain `Deny Read Or Deny Write`
- **Anchor Members**
- **Anchor Exclusion**
- **Dim1 Members**
- **Dim1 Exclusion**
- **Dim2 Members**
- **Dim2 Exclusion**
- **Dim X Members**
- **Dim X Exclusion**

Required Roles

Service Administrator

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

REST Resource

```
POST /HyperionPlanning/rest/{api_version}/applications/{application}/jobs
```

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Table 7-31 Import Cell-Level Security

Name	Description	Required	Default
jobType	Import Cell-Level Security or IMPORT_CELL_LEVEL_SECURITY	Yes	None
jobName	The name of the job to be used. This job name appears on the job console. Example: ImportCLSPlan1	No	Import Cell-Level Security Definitions
fileName	The name of the ZIP file containing the input Excel file. The file must be present in the Inbox. Before using this REST API, you can use the Upload REST API to upload the file.	Yes	None
errorFile	Optionally, create a separate error file for recording any errors that occur during the import process. If this is not specified, an error file is auto-generated with a name containing the user name, current date, and time stamp. For example, admin_ImportError_2020-02-26-04-34-45-420.txt. The file containing the error messages is stored in the Outbox. You can download it using the Download API . This API overrides any existing error file with the same name.	No	The file is auto-generated

For a sample URL, see the sample URL and payload in [Execute a Job](#)

Example 1: Imports cell-level security records from the input file

ImportCLSRecordsFile.zip.

```
{
  "jobType": "Import Cell-Level Security",
  "jobName": "ImportCLSJob",
  "parameters": {
    "fileName": "ImportCLSRecordsFile.zip"
  }
}
```

Example 2: Imports cell-level security records from the input file `ImportCLSRecordsFile.zip` and exports the error messages to the file `ImportCLSFileLog.txt`.

```
{
  "jobType": "Import Cell-Level Security",
  "jobName": "ImportCLSJob",
  "parameters": {
    "fileName": "ImportCLSRecordsFile.zip",
    "errorFile": "ImportCLSRecordsFileLog.txt"
  }
}
```

Export Cell-Level Security

This REST API exports cell-level security settings from Planning or Tax Reporting into a ZIP file that contains an Excel file. Cell-level security enables Service Administrators to restrict who can view data in the application by defining rules that remove read or write access to cells that a user would normally have access to due to their regular security.

The generated file is compressed, and the output is a ZIP file that is added to the Outbox. You can download the file using the Download REST API.

Note the following requirements for the format of the Excel file used with this REST API.

The exported Excel file contains two worksheets with these names:

1. Rules
2. Sub Rules

The `Rules` worksheet has the following column headings:

- **Name**
- **Position**
- **Description**
- **Enabled**
- **Valid Cubes** - This column contains either `All` or a list of comma-separate names of cubes, such as `Plan1, Plan2`
- **Anchor Dim Name**
- **Anchor Dimension Apply to Unselected Members**
- **Dim1**
- **Dim1 Required**
- **Dim2**
- **Dim2 Required**
- **Dim X Members**
- **Dim X Required**

The `Sub Rules` worksheet has the following column headings:

- **Name** - This column contains the names of the Rules from the first sheet
- **Users**

- **User Groups**
- **Restriction** - This column can contain Deny Read Or Deny Write
- **Anchor Members**
- **Anchor Exclusion**
- **Dim1 Members**
- **Dim1 Exclusion**
- **Dim2 Members**
- **Dim2 Exclusion**
- **Dim X Members**
- **Dim X Exclusion**

Required Roles

Service Administrator

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

REST Resource

```
POST /HyperionPlanning/rest/{api_version}/applications/
{application}/jobs
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Table 7-32 Export Cell Level Security

Name	Description	Required	Default
jobType	Export Cell-Level Security or EXPORT_CELL_LEVEL_SECURITY	Yes	None
jobName	The name of the job to be used. This job name appears on the job console. Example: ExportCLSDPlan1	No	Export Cell-Level Security Definitions
fileName	The name of the ZIP file that will be created to hold the Excel file containing cell-level security information. The file containing the exported data is stored in the Outbox.	Yes	None

Table 7-32 (Cont.) Export Cell Level Security

Name	Description	Required	Default
names	<p>Optionally, include a comma-separated list of cell-level security definitions in the application. Information from each definition is exported to a separate Excel file and then zipped. For example, the list could contain CLSDAccountPeriod,CLSDEntityPeriod,CLSDProductPeriod.</p> <p>If this parameter is not provided, all cell-level security records are exported.</p>	No	All records are exported

For a sample URL, see the sample URL and payload in [Execute a Job](#)

Example 1: Exports all cell-level security records to the file `ExportCLSDRecordsFile.zip`.

Sample Payload

```
{
  "jobType": "Export Cell-Level Security",
  "jobName": "ExportCellLevelSecurityJob",
  "parameters": {
    "fileName": "ExportCLSDRecordsFile.zip"
  }
}
```

Example 2: Exports three cell-level security records with names `CLSDAccountPeriod`, `CLSDEntityPeriod`, and `CLSDProductPeriod` to the file `Export3CLSDRecordsFile.zip`.

```
{
  "jobType": "Export Cell-Level Security",
  "jobName": "ExportCellLevelSecurityJob",
  "parameters": {
    "fileName": "Export3CLSDRecordsFile.zip",
    "names": "CLSDAccountPeriod,CLSDEntityPeriod,CLSDProductPeriod"
  }
}
```

Import Valid Intersections

This REST API imports valid intersections groups from a ZIP file that contains an Excel file with valid intersection definitions into a Financial Consolidation and Close, Planning, or Tax Reporting business process.

The Excel file must be present in the Inbox. You can use the Upload REST API to upload the file. Any rejected records are generated in an Excel file that is zipped and copied to the Outbox.

The following is a general explanation of the Excel file. The file contains two Excel worksheets:

1. **Rules** - defines the intersection group, dimensions included, and properties such as Unspecified Valid and Additional Dims Required
2. **Sub Rules** - provides member selections and exclusions

The **Rules** worksheet has the following column headings.

- **Name**
- **Position**
- **Description**
- **Enabled**
- **Valid Cubes** - This column can contain either `All` or a list of comma-separated names of cubes, such as `Plan1, Plan2`
- **Anchor Dim Name**
- **Anchor Dimension Apply to Unselected Members**
- **Dim1**
- **Dim1 Required**
- **Dim2**
- **Dim2 Required**
- **Dim X**
- **Dim X Required**

The **Sub Rules** worksheet must have the following column headings:

- **Name** - This column must contain the name of the Rule from the first worksheet
- **Users**
- **User Groups**
- **Restriction** - This column can contain `Deny Read Or Deny Write`
- **Anchor Members**
- **Anchor Exclusion**
- **Dim1 Members**
- **Dim1 Exclusion**
- **Dim2 Members**
- **Dim2 Exclusion**
- **Dim X Exclusion**

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
jobs

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Table 7-33 Import Valid Intersections

Name	Description	Required	Default
jobType	Import Valid Intersections or IMPORT_VALID_INTERSECTIONS	Yes	None
jobName	The name of the job to be used for this job execution, exactly as it is defined in the application. Example: ImportVIAccountPeriod	No	Import Valid Intersections
fileName	The name of the ZIP file containing the input Excel file. The file must be present in the Outbox. Before using this REST API, you can use the Upload REST API to upload the file.	Yes	None
errorFile	Optionally, identify the name of a text file for recording any errors that occur during the import process. If this is not specified, an error file is auto-generated with a name containing the user name, current date, and time stamp. For example, admin_ImportError_2020-02-26-04-34-45-420.txt. The file containing the error messages is stored in the Outbox. You can download it using the Download REST API.	No	The file name is auto-generated

For a sample URL, see the sample URL and payload in [Execute a Job](#)

Example 1: Imports valid intersections records from the input file `ImportVIRecordsFile.zip`.

```
{
  "jobType": "Import Valid Intersections",
  "jobName": "ImportVIJob",
  "parameters": {
    "fileName": "ImportVIRecordsFile.zip"
  }
}
```



```
}  
}
```

Example 2: Imports valid intersections records from the input file `ImportVIRecordsFile.zip` and exports the error messages to the file `ImportVIRecordsFileLog.txt`.

```
{  
  "jobType": "Import Valid Intersections",  
  "jobName": "ImportVIJob",  
    "fileName": "ImportVIRecordsFile.zip",  
    "errorFile": "ImportVIRecordsFileLog.txt"  
}  
}
```

Export Valid Intersections

This REST API exports valid intersection groups (certain cell intersections filtered by rules when users enter data or select runtime prompts) from Financial Consolidation and Close, Planning, or Tax Reporting business processes.

The output is a ZIP file that is added to the Outbox. You can download the file using the Download REST API.

Note the following requirements for the format of the Excel file used with this REST API.

The exported Excel file contains two worksheets with these names:

1. Rules
2. Sub Rules

The `Rules` worksheet has the following column headings:

- **Name**
- **Position**
- **Description**
- **Enabled**
- **Anchor Dim Name**
- **Anchor Dimension Apply to Unselected Members**
- **Dim1**
- **Dim1 Required**
- **Dim2**
- **Dim2 Required**
- **Dim X**
- **Dim X Required**

The `Sub Rules` worksheet has the following column headings:

- **Name** - This column contains the names of the Rules from the first worksheet

- **Anchor Members**
- **Anchor Exclusion**
- **Dim1 Members**
- **Dim1 Exclusion**
- **Dim2 Members**
- **Dim2 Exclusion**
- **Dim X Members**
- **Dim X Exclusion**

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
POST      /HyperionPlanning/rest/{api_version}/applications/{application}/
jobs
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Table 7-34 Export Valid Intersections

Name	Description	Required	Default
jobType	Export Valid Intersections or EXPORT_VALID_INTERSECTIONS	Yes	None
jobName	The name of the job to be used. This job name appears on the job console. Example: ExportVIRecords	No	Export Valid Intersections
fileName	The name of the ZIP file that will be created to hold the Excel file containing valid intersection records. The file containing the exported data is stored in the Outbox.	Yes	None

Table 7-34 (Cont.) Export Valid Intersections

Name	Description	Required	Default
names	<p>If provided, exports some of the records using the names in a comma-separated list of valid intersection record names. For example, the list could contain <code>VIAccountPeriod, VIEntityPeriod, VIProductPeriod</code>.</p> <p>If this parameter is not provided, all valid intersection records are exported.</p>	No	All records are exported

For a sample URL, see the sample URL and payload in [Execute a Job](#)

Sample Payload

Example 1: Exports all valid intersections records to the file `ExportVIRecordsFile.zip`.

```
{
  "jobType": "Export Valid Intersections",
  "jobName": "ExportVIJob",
  "parameters": {
    "fileName": "ExportVIRecordsFile.zip"
  }
}
```

Example 2: Exports three valid intersection records with names `VIAccountPeriod`, `VIEntityPeriod`, and `VIProductPeriod` to the file `Export3VIRecordsFile.zip`.

```
{
  "jobType": "Export Valid Intersections",
  "jobName": "ExportVIJob",
  "parameters": {
    "fileName": "Export3VIRecordsFile.zip",
    "names": "VIAccountPeriod,VIEntityPeriod,VIProductPeriod"
  }
}
```

Execute a Report Bursting Definition

You can execute bursting for a single report or book for more than one member of a single dimension, and publish a PDF or Excel output for each member.

The bursting definition must be present in the folder that you specify with the `burstingDefinitionName` parameter.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
jobs

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For parameters that are common to all jobs, see [Execute a Job](#).

Table 7-35 Execute a Report Bursting Definition

Name	Description	Required	Default
jobType	Execute Bursting Definition or EXECUTE_MR_BURST (both parameters are supported)	Yes	None
jobName	The name of the job to be used. This job name appears on the job console. Example: MonthlySalesBurstDev	No	Execute Bursting Definition
burstingDefinition Name	Bursting definition name with the complete path to where the bursting definition is stored. Example: /Library/Jan/ MonthlySalesBurstDef	Yes	None

For a sample URL, see the sample URL and payload in [Execute a Job](#)

Sample Payload

Example 1: Executes the bursting definition named `MonthlySalesBurstDev` that is present in the Library folder.

```
{
  "jobType": "Execute Bursting Definition",
  "jobName": "Execute MonthlySalesBurstDef",
  "parameters": {
    "burstingDefinitionName": "Library/MonthlySalesBurstDef"
  }
}
```

Example 2: Executes the bursting definition named `MonthlySalesBurstDev` that is present in the Reports subfolder under the Library folder.

```
{
  "jobType":"Execute Bursting Definition",
  "jobName":"Execute MonthlySalesBurstDef",
  "parameters": {
    "burstingDefinitionName":"Library/Reports/MonthlySalesBurstDef"
  }
}
```

Execute Job Code Samples

Example 7-1 Java Sample – `executeJob.java`

```
//
// BEGIN - Execute a Job (EXPORT_DATA, EXPORT_METADATA, IMPORT_DATA,
// IMPORT_METADATA, CUBE_REFRESH, ...)
//
public void executeJob(String jobType, String jobName, String
parameters) throws Exception {
    String urlString = String.format("%s/HyperionPlanning/rest/%s/
applications/%s/jobs", serverUrl, apiVersion, applicationName);
    JSONObject payload = new JSONObject();
    payload.put("jobName",jobName);
    payload.put("jobType",jobType);
    payload.put("parameters",new JSONObject(parameters));
    String response = executeRequest(urlString, "POST",
payload.toString());
    System.out.println("Job started successfully");
    getJobStatus(fetchPingUrlFromResponse(response, "self"),
"GET");
}
//
// END - Execute a Job (EXPORT_DATA, EXPORT_METADATA, IMPORT_DATA,
// IMPORT_METADATA, CUBE_REFRESH, ...)
//
```

Example 7-2 cURL Sample – `ExecuteJob.sh`

```
funcExecuteJob() {
    url="$SERVER_URL/HyperionPlanning/rest/$API_VERSION/
applications/$APP_NAME/jobs"
    encodedJobName=$(echo $2 | sed -f urlencode.sed)
    if [ ! -z "$3" ]; then

param="{\"jobType\": \"$1\", \"jobName\": \"$encodedJobName\", \"parameters
\":$3}"
        else

param="{\"jobType\": \"$1\", \"jobName\": \"$encodedJobName\"}"
        fi
    funcExecuteRequest "POST" $url $param
}
```

```

output=`cat response.txt`
status=`echo $output | jq '.status'`
if [ $status == -1 ]; then
    echo "Started executing job successfully"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 7-3 Groovy Sample – ExecuteJob.groovy

```

def executeJob(jobType, jobName, parameters) {
    def url = new URL(serverUrl + "/HyperionPlanning/rest/" + apiVersion + "/"
applications/" + appName + "/jobs");
    JSONObject payload = new JSONObject();
    try {
        if (parameters != null) {
            JSONObject params = new JSONObject();
            def args = parameters.split(';');
            for (int i = 0; i < args.length; i++) {
                if (args[i].indexOf("=") != -1) {
                    String[] param = args[i].split("=");
                    if
(param[0].equalsIgnoreCase("clearData")) {
parameters.put("clearData", Boolean.valueOf(param[1]));
                    }
                    else {
                        params.put(param[0], param[1]);
                    }
                }
            }
            payload.put("jobName", jobName);
            payload.put("jobType", jobType);
            payload.put("parameters", params);
        }
        else {
            payload.put("jobName", jobName);
            payload.put("jobType", jobType);
        }
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", payload);
    if (response != null) {
        getJobStatus(fetchPingUrlFromResponse(response, "self"), "GET");
    }
}
}

```

Retrieve Job Status

Polls the server to get the processing state for a job with a specified ID.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/jobs/{jobIdentifier}
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-36 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
applicationName	The name of the application	Path	Yes	None
jobIdentifier	The ID of the job	Path	Yes	None

Response

Parameters

The following table summarizes the response parameters.

Table 7-37 Parameters

Name	Description
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 = cancel pending; 3 = cancelled; 4 = invalid parameter; Integer.MAX_VALUE = unknown
details	Details about the job status, such as "Metadata import was successful" for metadata import
jobID	The ID of the job, such as 224
jobName	The name of the job, such as Refresh Database
descriptiveStatus	The status of the job, such as Completed or Error

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body for metadata import:

```
{
  "status": 0,
  "details": "Metadata import was successful",
  "jobId": 224,
  "jobName": "Import Account Metadata",
  "descriptiveStatus": "Completed",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/224",
    "action": "GET"
  }, {
    "rel": "job-details",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/224/details",
    "action": "GET"
  }
]}
}
```

The following shows an example of the response body when an error occurs during cube refresh:

```
{
  "status": 1,
  "details": "An error occurred while updating the relational database.",
  "jobStatus": "Error",
  "jobId": 145,
  "jobName": "Refresh Database",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/jobs/145",
    "action": "GET"
  }], {
    "rel": "job-details",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/145/details",
    "action": "GET"
  }
]}
}
```

The following shows an example of the response body when an error occurs during cube refresh:

```
{
  "status": 1,
  "details": "An error occurred while updating the relational database.",
  "jobStatus": "Error",
  "jobId": 145,
  "jobName": "Refresh Database",
```



```

    "links": [{
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/jobs/145",
      "action": "GET"
    }], {
      "rel": "job-details",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/test2/jobs/145/details",
      "action": "GET"
    }
  ]
}

```

Retrieve Job Status Details

Polls the server to get execution details for a Job with the specified Job ID. The job types for which details are returned by this service are: IMPORT_DATA, EXPORT_DATA, EXPORT_METADATA, and IMPORT_METADATA.

Supports paging for jobs of type IMPORT_DATA, IMPORT_METADATA, EXPORT_DATA, and EXPORT_METADATA using the offset and limit query parameters shown in the table.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/
{application}/jobs/{jobIdentifier}/details
```

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-38 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
jobIdentifier	The ID of the job, such as 224	Path	Yes	None

Table 7-38 (Cont.) Parameters

Name	Description	Type	Required	Default
msgType	Optionally, return messages for a particular message type. If no messageType is provided, returns messages of types ERROR, WARNING, and INFO.	Query	No	None
offset	For paging of jobs. Indicates the actual index from which the records are returned. It is 0 based.	Query	No	0
limit	For paging for jobs. Controls how many items to return. Defaults to 25 if not specified.	Query	No	25

Example Requests

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4app1/jobs/145/details
```

Optionally specifying messageType:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4app1/jobs/145/details?q={"messageType":"ERROR"}
```

Optionally specifying paging for jobs of type IMPORT_DATA and EXPORT_DATA with the offset and limit:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4app1/jobs/145/details?q={"messageType":"ERROR"}&offset=0&limit=5}
```

Response

The following table summarizes the response parameters.

Table 7-39 Parameters

Name	Description
items	Version of the API you are developing with
recordsRead	The name of the application
recordsRejected	The ID of the job for records rejected
recordsProcessed	The ID of the job for records processed
dimensionName	For paging of jobs. Indicates the actual index from which the records are returned. It is 0 based.
loadType	For paging for jobs. Controls how many items to return. Defaults to 25 if not specified.

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body for metadata import with `messageType = ERROR`:

```
{
  "items": [{
    "links": [{
      "rel": "child-job-details",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/test2/jobs/224/childjobs/12/details?
limit=10&q=%7BmessageType:ERROR%7D&offset=10",
      "action": "GET"
    }],
    "recordsRead": 8,
    "recordsRejected": 0,
    "recordsProcessed": 8,
    "dimensionName": "Entity",
    "loadType": "Metadata Import"
  }, {
    "links": [{
      "rel": "child-job-details",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/test2/jobs/224/childjobs/13/details?
limit=10&q=%7BmessageType:ERROR%7D&offset=10",
      "action": "GET"
    }],
    "recordsRead": 2,
    "recordsRejected": 0,
    "recordsProcessed": 2,
    "dimensionName": "Job",
    "loadType": "Metadata Import"
  }],
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/test2/jobs/224/details?
limit=10&q=%7BmessageType:ERROR%7D&offset=10",
    "action": "GET"
  }],
}
```

Retrieve Child Job Status Details

Certain types of jobs, such as metadata import and export, create child jobs for each dimension being exported or imported. This service can be used to get the execution details for the child Job with the specified ID.

The job types for which child details are returned by this service are `IMPORT_METADATA` and `EXPORT_METADATA`. Supports paging for jobs of type `IMPORT_DATA` and `EXPORT_DATA` using the offset and limit query parameters shown in the table.

Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/jobs/
{jobIdentifier}/childjobs/{childJobIdentifier}/details
```

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-40 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
applicationName	The name of the application	Path	Yes	None
jobIdentifier	The ID of the job, such as 224	Path	Yes	None
childJobIdentifier	The ID of the child job	Path	Yes	None
childJobID	The ID of the child job, such as 8	Path	Yes	None
q				
messageType	Optionally, return messages for a particular message type. If no messageType is provided, returns messages of types ERROR, WARNING, and INFO.	Query	Yes	None
offset	For paging of jobs. Indicates the actual index from which the records are returned. It is 0 based.	Query	No	0
limit	For paging of jobs. Controls how many items to return. Defaults to 25 if not specified.	Query	No	25

Example Requests

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/jobs/145/childjobs/123/details
```

Optionally specifying messageType:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/jobs/145/ childjobs/123/details?
q={"messageType":"ERROR"}
```

Optionally specifying paging for jobs of type IMPORT_METADATA and EXPORT_METADATA with an offset and limit:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/jobs/145/ childjobs/123/details?
q={"messageType":"ERROR"}&offset=0&limit=5
```

Response

The following table summarizes the response parameters.

Table 7-41 Parameters

Name	Description
items	The number of records read, such as 8
msgType	Message type, such as INFO
msgCategory	Message category, such as Argument parsing
msgText	Message text

Supported Media Types: application/json

Example of Response Body

The following example shows a response body with messageType = INFO, offset=5, limit=5. Notice the prev and next links.

```
{
  "items": [{
    "msgType": "INFO",
    "msgCategory": "Argument parsing",
    "msgText": "The column alias mapping list specified with
the /C2A switch did not match a key in the Command Properties file
\"null\" so it will be used as the mapping directly:
\"(<ignoreUndefined>,@Plan*)\"."
  }, {
    "msgType": "INFO",
    "msgCategory": "Unclassified",
    "msgText": "Header record fields: Entity, Parent, Alias:
Default, Alias: SLAliases, Valid For Consolidations, Data Storage, Two
Pass Calculation, Description, Formula, UDA, Smart List, Data Type,
Hierarchy Type, Enable for Dynamic Children, Number of Possible Dyn..."
  }, {
    "msgType": "INFO",
    "msgCategory": "Dimension, member, or cube retrieval",
    "msgText": "Located and using \"Entity\" dimension for loading
data in \"Test2\" application."
  }, {
    "msgType": "INFO",
    "msgCategory": "Unclassified",
    "msgText": "HspOutlineLoad::dateFormatSpecified is true,
SessionHalDateFormat stored on session: M/d/yy, sessionId: 759992870"
  }, {
    "msgType": "INFO",
    "msgCategory": "Dimension, member, or cube retrieval",
    "msgText": "Load dimension \"Entity\" has been unlocked
successfully."
  }
  ],
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-"
  }
  ]
}
```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/224/childjobs/12/details?limit=5&offset=5",
    "action": "GET"
  }, {
    "rel": "prev",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/224/childjobs/12/details?offset=0&limit=5",
    "action": "GET"
  }, {
    "rel": "next",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/test2/jobs/224/childjobs/12/details?offset=10&limit=5",
    "action": "GET"
  }
}],
}

```

Working with Members

You can get and add members using a set of REST resources, as summarized below.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 7-42 Working with Members

Task	Request	REST Resource
Add Member	POST	/HyperionPlanning/rest/{api_version}/applications/{application}/dimensions/{dimensionname}/members
Get Member	GET	/HyperionPlanning/rest/{api_version}/applications/{application}/dimensions/{dimension}/members/{member}

Add Member

Adds a new member to the application outline in the specified dimension and plan type and under the specified parent member.

Note:

Prerequisite: The parent member must be enabled for dynamic children and a cube refresh must have happened after the parent was enabled.

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
dimensions/{dimname}/members

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-43 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application to which to add a member	Path	Yes	None
dimname	Name of the dimension to which to add a member	Path	Yes	None

Example URL and Payload

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/dimensions/Entity/members`

```
{"memberName":"North America","parentName":"Enterprise Global"}
```

Response

Payload Parameters:

The following table summarizes the payload parameters.

Table 7-44 Parameters

Name	Description
name	Name of the member, such as North America
children	Whether the member has children
description	Description of the member
parentName	Name of the parent, such as Enterprise Global
dataType	Data type, if available
objectType	Type of object
dataStorage	Storage attribute for the member, such as STOREDATA
dimName	Dimension name
twoPass	Boolean value to indicate whether the member has the Two-Pass Calculation associated attribute
instance	Information about the instance
type	Type of member
detail	Detailed information in case of error
status	Request status, such as 400

Table 7-44 (Cont.) Parameters

Name	Description
errorPath	Path of the error, if available
title	Error title, if available
errorCode	Error code, if available
errorDetails	Error details, if available
message	Message text
localizedMessage	Localized message, if available

Example of Response Body

Sample response body where the member is added successfully

```
{
  "name": "North America",
  "children": null,
  "description": null,
  "parentName": "Enterprise Global",
  "dataType": "UNSPECIFIED",
  "objectType": 33,
  "dataStorage": "STOREDATA",
  "dimName": "Entity",
  "twoPass": false,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/Vision/dimensions/Entity/members/North%20America",
    "action": "GET"
  }]
}
```

Sample response when an error occurs when adding the member:

```
{
  "detail": "Error occurred adding member. Unable to find parent
<Enterprise GlobalX> defined for a dynamic member.",
  "status": 400,
  "message": "com.hyperion.planning.HspRuntimeException: Error occurred
adding member. Unable to find parent <Enterprise GlobalX> defined for a
dynamic member.",
  "localizedMessage": "com.hyperion.planning.HspRuntimeException: Error
occurred adding member. Unable to find parent <Enterprise GlobalX> defined
for a dynamic member."
}
```

Get Member

Gets the specified member's properties.

Required Roles

Service Administrator

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/
dimensions/{dimname}/members/{member}
```

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-45 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to get member properties	Path	Yes	None
dimname	Name of the dimension for which to get member properties	Path	Yes	None
member	Name of the member for which to get member properties	Path	Yes	None

Request Payload:

The following table summarizes the payload parameters.

Table 7-46 Parameters

Name	Description
name	Name of the member, such as North America
children	Whether the member has children
description	Description of the member
parentName	Name of the parent, such as Enterprise Global
dataType	Data type, if available
objectType	Type of object
dataStorage	Storage attribute for the member, such as STOREDATA
dimName	Dimension name
twoPass	Boolean value to indicate whether the member has the Two-Pass Calculation associated attribute

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/dimensions/Entity/North
America
```

```
{"memberName":"North America","parentName":"Enterprise Global"}
```

Example of Response Body

Sample response body where the member is added successfully

```
{
  "name": "North America",
  "children": null,
  "description": null,
  "parentName": "Enterprise Global",
  "dataType": "UNSPECIFIED",
  "objectType": 33,
  "dataStorage": "STOREDATA",
  "dimName": "Entity",
  "twoPass": false,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/Vision/dimensions/Entity/members/North%20America",
    "action": "GET"
  }]
}
```

Get Applications

This REST API returns a list of applications to which the specified user is assigned.

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/{api_version}/applications

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-47 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 7-48 Parameters

Name	Description
Items	A list of applications
name	Application name
type	Product type. Possible values: HFM, HP
adminMode	Indicates if the application's login level is set to Administrators. Returns a Boolean value where <code>true</code> indicates that the login level for the application is set to Administrators and <code>false</code> indicates that the login level is set to All Users.

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "HP",
  "items": [
    {
      "appType": "PBCS",
      "webBotDetails": "null",
      "helpServerUrl": "https://www.oracle.com",
      "workspaceServerUrl": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com",
      "appStorage": "Multidim",
      "adminMode": false,
      "unicode": true,
      "name": "Vision",
      "type": "HP"
      "adminMode": "true"
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications",
      "action": "GET"
    }
  ]
}
```

Manage Planning Units

You can manage planning units using a set of REST resources, as summarized below.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

For detailed information on managing planning units, see [Managing Approvals](#).

Note: The manage planning unit resources use the parameters `puIdentifier` and `puhIdentifier`:

- `puIdentifier`: Planning unit identifier
- `puhIdentifier`: Planning unit hierarchy identifier

Use the following format for these parameters:

- `puIdentifier`:
"scenarioName"::"versionName"::"pmMember"
- `pmMember`:
"Entity: SecondaryMember"
- `puhIdentifier`
"scenarioName"::"versionName"

Table 7-49 Managing Planning Units

Task	Request	REST Resource
List All Planning Units	POST	/HyperionPlanning/rest/{version}/applications/{application}/planningunits?q={"scenario":"scenarioName","version":"versionName"}&offset=10&limit=10
Get Planning Unit History and Annotations	GET	/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits?q={"scenario":"scenarioName","version":"versionName"}&offset={offset}&limit={limit}
Get a Planning Unit Owner Photo	GET	/HyperionPlanning/rest/{api_version}/applications/{application}/users/{userId}/photo
Get Planning Unit Promotional Path	GET	/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits/{puIdentifier}/promotionpath
Get Available Planning Unit Actions	GET	/HyperionPlanning/rest/{api_version}/applications/{applicationName}/planningunits/{puhIdentifier}/availableactions
Get Filters with All Possible Values	GET	/HyperionPlanning/rest/{api_version}/applications/{application}/pufilters
Change Planning Unit Status	POST	/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits/{puhIdentifier}/actions

List All Planning Units

Returns a list of planning units for the specified application and owned by the user initiating the REST API. (Note that this does not return all planning units for all applications and users.)

Paging is supported if the optional offset and limit parameters are provided.

Required Roles

Service Administrator

REST Resource

POST

```
/HyperionPlanning/rest/{version}/applications/{application}/planningunits?
q={"scenario":"scenarioName","version":"versionName"}&offset=10&limit=10
```

Request

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the client request.

Table 7-50 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
q	Paging options for planning units. Possible values are described in the following rows.	Query	No	Limit = 25
scenario	Scenario for the application; required	Query	No	None
version	Version for the application; required	Query	No	None
offset	Indicates the actual index from which the records are returned; 0 based.	Query	No	1
limit	Controls how many items to return; defaults to 25 if not specified.	Query	No	25

Request Payload:

The following table summarizes the payload parameters.

Table 7-51 Parameters

Name	Description	Type	Required
filter	Name, type, and values to filter on. Example: filter={name:"SubStatus",type:3,values:[0,4]}	Payload	No

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/{version}/applications/{application}/planningunits?
q={"scenario":"scenarioName","version":"versionName"}&offset=10&limit=10
```

Example without filters:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/planningunits?
q={"scenario":"Forecast","version":"BU Version_1"}
```

Example with two filters, multiple values provided:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/planningunits?
q={"scenario":"Forecast","version":"BU Version_1"}
```

Payload:

```
filter={name:"Status",type:4,values:
[2,5]}&filter={name:"SubStatus",type:3,values:[0,4]}
```

Response

Supported Media Types: application/json

Parameters:

Table 7-52 Parameters

Name	Description
name	Planning unit name, such as Marketing
value	Planning unit value
owner	Planning unit owner, such as Admin
version	Planning unit version, such as BU Version_1
entity	Planning unit entity, such as Marketing
status	Planning unit status, such as Under Review
scenario	Planning unit scenario, such as Forecast
Formatted value	Formatted value, if any
puName	Planning unit name, such as Marketing
subStatus	Planning unit substatus
secMember	Secondary dimension member, if any
puAlias	Planning unit alias, such as Marketing
versionAlias	Version alias, if any

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "name": null,
    "value": -1.0,
    "owner": "admin",
    "version": "BU Version_1",
    "entity": "Marketing",
    "status": "Under Review",
    "scenario": "Forecast",
    "formattedValue": "",
    "puName": "Marketing",
```

```

    "subStatus": "",
    "secMember": null,
    "puAlias": "Marketing",
    "scenarioAlias": null,
    "versionAlias": null,
    "puId": 50410,
    "links": [{
      "rel": "promotion-path",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
%22Forecast%22:%22BU%20Version_1%22:%22Marketing%22:%22%22/
promotionpath",
      "action": "GET"
    }, {
      "rel": "annotations-and-history",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
%22Forecast%22:%22BU%20Version_1%22:%22Marketing%22:%22%22/
historyandannotations?q=%7B%22annotSeq%22:-1,%22logSeq%22:-1%7D",
      "action": "GET"
    }, {
      "rel": "actions",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
%22Forecast%22:%22BU%20Version_1%22/actions",
      "action": "POST",
      "data": {
        "pmMembers": "Marketing"
      }
    }, {
      "rel": "change-status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
%22Forecast%22:%22BU%20Version_1%22:%22Marketing%22:%22%22/
actions/6",
      "action": "POST",
      "data": {
        "pmMembers": "Marketing",
        "comments": "comments"
      }
    }
  ]],
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits?
q=%7Bscenario:%22Forecast%22,version:%22BU+Version_1%22%7D",
    "action": "POST",
    "data": {
      "filter": [{

```

```

        "name": "Status",
        "type": 4,
        "values": [2, 5],
        "defIndex": 0
    }, {
        "name": "SubStatus",
        "type": 3,
        "values": [0, 4],
        "defIndex": 0
    }
  ]
  "type": "HP"
}

```

Get Planning Unit History and Annotations

Returns a merged list of history and annotations for the planning unit that the requesting user owns for the specified Scenario, Version, and PM Member.

If both `annotSeq` and `logSeq` are `< 0`, parent level nodes are returned. If `annotSeq` or `logSeq` is provided, the replies to that annotation or history are returned respectively.

If both `annotSeq` and `logSeq` are `< 0`, parent level nodes are returned. If `annotSeq` or `logSeq` is provided, the replies to that annotation or history are returned respectively.

Required Roles

Service Administrator

REST Resource

```

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
planningunits/{puIdentifier}/historyandannotations?
q={annotSeq=-1,logSeq=-1}&offset=10&limit=10

```

Request

Supported Media Types: `application/x-www-form-urlencoded`

Parameters:

The following table summarizes the client request.

Table 7-53 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Version of the API you are developing with	Path	Yes	None
<code>application</code>	The name of the application	Path	Yes	None
<code>scenario</code>	Scenario for the application; required	Path	Yes	None
<code>version</code>	Version for the application; required	Path	Yes	None
<code>pmMember</code>	Entity; secondary member	Path	Yes	None
<code>offset</code>	Indicates the actual index from which the records are returned; 0 based.	Query	No	1

Table 7-53 (Cont.) Parameters

Name	Description	Type	Required	Default
limit	Controls how many items to return; defaults to 25 if not specified.	Query	No	25

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4app1/planningunits/Forecast::"BU
Version_1"::Marketing::/historyandannotations?q={annotSeq:-1,logSeq:-1}
```

Filter can include name, type, and values to filter on. For example:

```
filter={name:"SubStatus",type:3,values:[0,4]}
```

Response

Supported Media Types: application/x-www-form-urlencoded

Parameters:

Table 7-54 Parameters

Name	Description
comment	Comment entered by the planning unit owner when performing an action
hasHistory	True if the planning unit has history
logSeq	Sequence of the action performed on the planning unit
staticImage	Whether a static image exists for this note
authorImagePath	The path to the user image for the user who performed the action
commentTitle	The author name and the action the author performed
commentDate	The date when the action was performed or the annotation was added
commentSubTitle	Processing state of the planning unit when the action was performed
parentAnntSeq	Sequence of the annotation or the parent annotation added to the planning unit
isChildNode	true if this is a reply to an annotation

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "comment": "Enough justification provided, Approving it.<p></p>",
    "hasHistory": false,
    "logSeq": -1,
    "staticImage": true,
```

```
    "authorImagePath": "/Images/GhostUser.png",
    "commentTitle": "admin",
    "commentDate": "8/22/14 3:41 PM",
    "commentSubTitle": "",
    "parentAnntSeq": 1,
    "isChildNode": false,
    "links": [{
      "rel": "annotation-replies",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/planningunits/
%22Forecast%22::%22BU%20Version_1%22::%22Marketing%22::/
historyandannotations?q=%7B%22annotSeq%22:1,%22logSeq%22:-1%7D",
      "action": "GET"
    }]
  }, {
    "comment": "",
    "hasHistory": true,
    "logSeq": 2,
    "staticImage": true,
    "authorImagePath": "/Images/GhostUser.png",
    "commentTitle": "Originate by admin",
    "commentDate": "4/22/14 12:26 PM",
    "commentSubTitle": "Under Review",
    "parentAnntSeq": -1,
    "isChildNode": false,
    "type": "HP",
    "links": [{
      "rel": "annotation-replies",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/planningunits/
%22Forecast%22::%22BU%20Version_1%22::%22Marketing%22::/
historyandannotations?q=%7B%22annotSeq%22:-1,%22logSeq%22:2%7D",
      "action": "GET"
    }]
  }, {
    "comment": "",
    "hasHistory": true,
    "logSeq": 1,
    "staticImage": true,
    "authorImagePath": "/Images/GhostUser.png",
    "commentTitle": "Originate by admin",
    "commentDate": "4/22/14 12:26 PM",
    "commentSubTitle": "Under Review",
    "parentAnntSeq": -1,
    "isChildNode": false,
    "type": "HP",
    "links": [{
      "rel": "annotation-replies",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/planningunits/
%22Forecast%22::%22BU%20Version_1%22::%22Marketing%22::/
historyandannotations?q=%7B%22annotSeq%22:-1,%22logSeq%22:1%7D",
```

```

        "action": "GET"
      ]]
    }],
    "type": "HP",
    "links": [{
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
%22Forecast%22::%22BU%20Version_1%22::%22Marketing%22::%22%22/
historyandannotations?q=%7B%22annotSeq%22:-1,%22logSeq%22:-1%7D",
      "action": "GET"
    }],
    "type": "HP",
  }
}

```

Get a Planning Unit Owner Photo

Returns an image for the requested planning unit owner if a photo is uploaded for the owner.

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/{api_version}/applications/{application}/users/{userId}/photo

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-55 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
userID	The identifier of the user for whom to retrieve a photo	Path	Yes	

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4appl/users/5000100/photo

Response

Supported Media Types: In case of success, returns `application/octet-stream`. In case of error, returns `application/json`.

Error Response:

Table 7-56 Parameters

Name	Description
detail	Detail about the status of the planning unit photo for this user. For example: PU photo not available, make sure that a valid user identifier is provided.
status	HTTP status, such as 400.
message	Informational message about the status of the photo for this user.
localizedMessage	A localized informational message about the status.

Example of Error Response Body

The following shows an example of the response body in JSON format.

```
{
  "detail": "PU photo not available, make sure the a valid user identifier
is provided.",
  "status": 400,
  "message": "java.lang.RuntimeException: PU photo not available, make
sure the a valid user identifier is provided.",
  "localizedMessage": "java.lang.RuntimeException: PU photo not available,
make sure the a valid user identifier is provided."
}
```

Get Planning Unit Promotional Path

Returns a list of promotion path nodes for a given application, user, and planning unit. The planning unit is identified by the provided scenario, version, and PM member. The list can have up to three nodes: the node before the current location, the node at the current location, and the one after the current location. If the planning unit is at the starting location or the last location in the path, only two nodes are returned.

Required Roles

Service Administrator

REST Resource

GET `/HyperionPlanning/rest/{api_version}/applications/{application}/planningunits{puIdentifier}/promotionpath`

Request

Parameters:

The following table summarizes the client request.

Table 7-57 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
puIdentifier	The name of the planning unit, such as Sales	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/planningunits/Forecast::"BU
Version_1"::Dev/promotionpath
```

Response

Supported Media Types: application/json

Parameters:

Table 7-58 Parameters

Name	Description
Items	Planning unit promotional path information
name	Name of the planning unit
ownerType	Planning unit owner type
group	Returns whether the owner is a group of users or an individual users, true or false
staticImage	Returns whether the image is a static manage, true or false
nodeImagePath	Path to the planning unit owner photo
ownerName	Name of the planning unit owner
currentLoc	Returns if this is the current location of the planning unit, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "name": "ent_111: Regular Coke",
    "ownerType": 0,
    "group": false,
    "staticImage": true,
    "nodeImagePath": "../ui_themes/tadpole/images_product/pm/75X89/
PUOwner.png",
    "ownerName": "Planner1",
    "currentLoc": true
  }, {
    "name": "Total Entity",
```

```

    "ownerType": 0,
    "group": false,
    "staticImage": false,
    "nodeImagePath": "v3/applications/PS4appl/puphoto?appOwner=50001",
    "ownerName": "admin",
    "currentLoc": false
  }},
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/planningunits/Forecast::%22BU%20Version_1%22::Dev/
promotionpath",
    "action": "GET"
  }],
}

```

Get Available Planning Unit Actions

Returns a list of the next set of applicable actions available for the planning units, consisting of the specified scenario, version, and PM Members (Entity: Secondary member) that are owned by the requesting user.

Required Roles

Service Administrator

REST Resource

POST

/HyperionPlanning/rest/{api_version}/applications/{application}/
planningunits{puIdentifier}/availableactions

Request

Parameters:

The following table summarizes the client request.

Table 7-59 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
puIdentifier	The name of the planning unit, such as Sales	Path	Yes	None
q	Optionally, return limited or full approvals functionality. Options are listed here.	Query	No	None
0	Returns limited approvals functionality - useful for mobile clients	Query	No	None
1	Returns full approvals functionality; default is 1.	Query	No	None

URL and Payload Examples

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/planningunits/Forecast::"BU
Version_1"/availableactions?q={"options":1}
```

Payload examples:

```
pmMembers=pmMemberNames
```

```
pmMembers=Dev,Marketing
```

Response

Supported Media Types: application/json

Parameters:

Table 7-60 Parameters

Name	Description
Items	Planning unit available actions
actionId	ID of the action
Name	Name of the action

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [
    {
      "actionId": 6,
      "name": "Promote"
    },
    {
      "actionId": 3,
      "name": "Sign Off"
    },
    {
      "actionId": 1,
      "name": "Reject"
    },
    {
      "actionId": 7,
      "name": "Delegate"
    },
    {
      "actionId": 8,
      "name": "Take Ownership"
    },
    {
      "actionId": 9,
      "name": "Originate"
    },
    {
      "actionId": 10,
      "name": "Freeze"
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4_HP2/planningunits/
```

```

Current::%22BU%20Version_1%22/availableactions?q=%7Boptions:1%7D",
  "action": "GET",
  "data": {
    "pmMembers": "ent_111: Regular Coke"
  }
}],
}

```

Get Filters with All Possible Values

Returns all filter types with all possible values by which users can filter planning units for a given application. For every value, there is a label (in the client locale) representation and an integer value. The labels are shown to end users to pick from, but whenever possible, the client should submit the integer value that is unique to the server. Every application supports several types of filters that are indicated by the type field.

Name is optional. The `defIndex` is the index in the value array for the default selection value.

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/{api_version}/applications/{application}/pufilters

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-61 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
q	Optionally, return filters for limited or full functionality. Options are listed here.	Query	No	None
0	Returns filters for limited Approvals functionality - useful for mobile clients.	Query	No	None
1	Returns filters for full Approvals functionality; default is 1.	Query	No	None

Example URL

```

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/pufilters?q={"options":"0"}

```

Response

Supported Media Types: application/json

Parameters:

Table 7-62 Parameters

Name	Description
Names	Planning unit available actions
aliases	Aliases for values to be displayed instead of labels if user preference is defined as such
name	Name for the filter
type	Type of the filter
values	Integer values; usually the client will submit this value to indicate the selected filter
labels	Labels for values to be displayed in the client for the filter
defIndex	Index of the value to be displayed as the default value

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "aliases": ["Forecast"],
    "name": "Scenarios",
    "type": 1,
    "values": [50218],
    "labels": ["Forecast"],
    "defIndex": 0
  }, {
    "aliases": ["BU Version_1"],
    "name": "Versions",
    "type": 2,
    "values": [1500],
    "labels": ["BU Version_1"],
    "defIndex": 0
  }, {
    "aliases": null,
    "name": "SubStatus",
    "type": 3,
    "values": [0, 1, 2, 3, 4, 10008, 10009, 10000],
    "labels": ["", "Processing", "Aborted", "Validating", "No
Additional Approval Required", "Invalid Data", "Additional Approval
Required", "Failed"],
    "defIndex": 0
  }, {
    "aliases": null,
    "name": "Status",
    "type": 4,
    "values": [2, 3, 4, 5, 6],
    "labels": ["Under Review", "Approved", "Signed Off", "Not
Signed Off", "Frozen"],
    "defIndex": 0
  }
],
}
```

```

    "links": [{
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/pufilters?q=%7Boptions:%22%22%7D",
      "action": "GET"
    }],
  }

```

Change Planning Unit Status

Changes the status of the planning units consisting of the specified scenario, version, and PM Members (Entity: Secondary member) that are owned by the requesting user. An error will display if the planning units belong to same hierarchy but different levels, or if the statuses for the planning units are not the same.

Supported actions for limited approvals functionality are: "PROMOTE" (6), "SIGN_OFF" (3), "APPROVE" (2), "DELEGATE" (7), "TAKE_OWNERSHIP" (8), "ORIGINATE" (9), "FREEZE" (10)

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
planningunits/{puIdentifier}/actions

Request

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the client request.

Table 7-63 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None

Example URL and Payload

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/PS4appl/planningunits/Forecast::"BU
Version_1"/actions

Payload

actionId=actionId&pmMembers=pmMemberNames&comments=comments

Response

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body in JSON format.

```

{
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/PS4appl/planningunits/
Forecast::%22BU%20Version_1%22/actions",
    "action": "POST",
    "data": {
      "pmMembers": "\"Dev\"",
      "action": "PROMOTE",
      "comments": "\"Promoting the PU\""
    }
  }]
}

```

Get User Preferences

Returns the requesting user's display preferences.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/{api_version}/applications/{application}/userpreferences

Request

Parameters:

The following table summarizes the client request.

Table 7-64 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
userpreferences	The requesting user's display preferences	path	Yes	None

Example URL

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/PS4appl/userpreferences`

Response

Supported Media Types: application/json

Table 7-65 Parameters

Name	Description
decimal separator	Preference for decimal separator
scale	Preference for scale
thousandsSeparator	Preference for the thousands separator
thousandsSeparator	Preference for the style of negative numbers
negativeStyle	Preference for the minimum precision
minPrecision	Preference for the minimum precision
maxPrecision	Preference for the maximum precision
showPUAlias	Preference for showing the planning unit alias
currSymbol	Preference for the currency symbol
type	The type of application, such as HP

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "decimalSeparator": "",
  "scale": 0,
  "thousandsSeparator": "",
  "negativeStyle": 255,
  "minPrecision": 0,
  "maxPrecision": 0,
  "showPUAlias": false,
  "currSymbol": "",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/PS4appl/userpreferences",
    "action": "GET"
  }]
}
```

Working with Data Slices

You can import, export, and clear data slices, as summarized here. Note that attribute dimensions are not supported in the payload.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 7-66 Working with Data Slices

Task	Request	REST Resource
Import Data Slice	POST	/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/importdataslice
Export Data Slice	POST	/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/exportdataslice
Clear Data Slice	POST	/HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/cleardataslice

Import Data Slices

Can be used to import data given a JSON data grid with a point of view, columns, and one or more data rows. Data will be imported only for cells that the user has read-write access to. Imports data of types Text, Date and Smart List along with numeric data. Returns JSON with details on the number of cells that were accepted, the number of cells that were rejected, and the first 100 cells that were rejected. You can set custom parameters to view rejected cells to understand the reason for the rejection.

Required roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/importdataslice

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-67 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to import the data slice	Path	Yes	None
plantype	Name of the plan type for which to import the data slice	Path	Yes	None

Example URL and Payload:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/Vision/plantypes/plan1/importdataslice
```

Payload Parameters

The Payload is JSON with the following parameters.

Table 7-68 Parameters

Name	Description
dataGrid	JSON data grid
aggregateEssbaseData	<p>True or false.</p> <p>If true, the values being saved will be added to the existing values. Only numeric values can be aggregated. Cells with Smart list, Text and Date data types will be rejected.</p> <p>If false, the data values will be overwritten. A value of "#missing" will clear the cell value as shown in the example. The default is false.</p> <p>Note: Values provided in the "data" section of the JSON payload will be used even for cells with supporting details provided. For cells with supporting details, make sure the total calculated for the incoming supporting details matches the value provided in the row "data" section.</p> <p>See the following table for examples.</p>
cellNotesOption	<p>Possible values are: "Overwrite", "Append", and "Skip".</p> <ul style="list-style-type: none"> "Overwrite": The existing cell notes will be overwritten. An empty array for cell notes [] will indicate deletion of existing cell notes. A value of "null" will leave the existing cell notes intact. "Append": New cell notes will be appended to existing cell notes. "Skip": Cell notes will not be processed.
dateFormat	Date format used in the input data grid. Valid formats are: "MM-DD-YYYY", "DD-MM-YYYY", "YYYY-MM-DD", "MM/DD/YYYY", "DD/MM/YYYY", "YYYY/MM/DD"
strictDateValidation	Optionally, influence how Date cell values are validated. When set to true, date values are validated against the dateFormat specified in the payload and are rejected if the format for the value does not conform to the dateFormat. If set to false, date values are interpreted more leniently. Default is true.
customParams	
postDataImportRuleNames	Optionally, provide the post data import rule names. This is primarily used by Data Management for planners. Default is false.
includeRejectedCells	Optionally, indicate if the response should include the first 100 rejected cells. Default is true.
includeRejectedCellsWithDetails	Optionally, indicate if the response should include the reasons why cells are rejected. Default is false.

Table 7-69 Import Data Slice Examples

Source Cell	Target Cell	Resulting Target Cell
Supporting Detail (SD)	#missing	SD
SD	Value	Add SD value to the existing value, do not add SD
Value	SD	Delete SD, add Value to the existing value
SD1	SD2	Delete SD2, add SD value to the existing value, do not add SD1

Sample payload:

```
{
  "aggregateEssbaseData":true,
  "cellNotesOption":"Overwrite",
  "dateFormat":"DD/MM/YYYY",
  "strictDateValidation":true,
  "dryRun": true,
  "customParams":{
    "PostDataImportRuleNames":"Post data rule 1, \"post, data rule
2\"",
    "IncludeRejectedCells":true,
    "IncludeRejectedCellsWithDetails":true
  },
}
```

Response

Supported Media Types: application/json

JSON Output

The rejected cells consist of cells that the user does not have read-write access to; cells where row or column member names are invalid and do not exist; cells where the data is invalid (for example, an invalid Smart List value); and cells that are non-numeric (Smart List, Text, or Date type) with data when `aggregateEssbaseData` is set to true.

```
{
  "numAcceptedCells": 3,
  "numRejectedCells": 9,
  "rejectedCells": ["[BaseData, FY15, Plan, Working, 410, P_160, Jan,
Project Number]", "[BaseData, FY15, Plan, Working, 410, P_160, Feb,
Project Number]", "[BaseData, FY15, Plan, Working, 410, P_160, Mar,
Project Number]", "[BaseData, FY15, Plan, Working, 410, P_160, Jan,
Request Date]", "[BaseData, FY15, Plan, Working, 410, P_160, Feb, Request
Date]", "[BaseData, FY15, Plan, Working, 410, P_160, Mar, Request Date]",
"[BaseData, FY15, Plan, Working, 410, P_160, Jan, Project Type]",
"[BaseData, FY15, Plan, Working, 410, P_160, Feb, Project Type]",
"[BaseData, FY15, Plan, Working, 410, P_160, Mar, Project Type]"],
}
```

Export Data Slices

Can be used to export data for a specified region. The exported data will be in the form of a JSON grid with pov, columns, and 0 or more data rows. Data will be exported only for cells for which the user has read-write access.

Required Roles

Service Administrator

REST Resource

```
POST /HyperionPlanning/rest/{api_version}/applications/{application}/
plantypes/{plantype}/exportdataslice
```

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-70 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to export the data slice	Path	Yes	None
plantype	Name of the plan type for which to export the data slice	Path	Yes	None

Example URL and payload:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/plantypes/plan1/exportdataslice
Payload Parameters
```

The Payload is JSON with the following parameters.

Table 7-71 Parameters

Name	Description
gridDefinition	JSON grid definition to define the region
exportPlanningData	True or false. Optionally, you can provide the parameter exportPlanningData, which, when set to true, will export supporting details and cell notes along with Essbase numeric data. Default is false.
suppressMissingBlocks	True or false. Optionally, you can set suppressMissingBlocks to true to suppress blocks with missing data.
suppressMissingRows	True or false. Optionally, you can set suppressMissingRows to true to suppress rows with missing data.
suppressMissingColumns	True or false. Optionally, you can set suppressMissingColumns to true to suppress rows with missing data.

Supported Media Type: application

Sample Payload

Example 1: Providing dimension names as follows in the gridDefinition is recommended and is more efficient.

```
{
  "exportPlanningData": false,
  "gridDefinition": {
    "suppressMissingBlocks": true,
```



```
"pov": {
  "dimensions": [
    "HSP_View",
    "Year",
    "Scenario",
    "Version",
    "Entity",
    "Product"
  ],
  "members": [
    [
      "BaseData"
    ],
    [
      "FY15"
    ],
    [
      "Plan"
    ],
    [
      "Working"
    ],
    [
      "410"
    ],
    [
      "P_160"
    ]
  ]
},
"columns": [
  {
    "dimensions": [
      "Period"
    ],
    "members": [
      [
        "IDescendants(Q1)"
      ]
    ]
  },
  {
    "dimensions": [
      "Period"
    ],
    "members": [
      [
        "IDescendants(Q2)"
      ]
    ]
  }
],
"rows": [
  {
    "dimensions": [
```

```

        "Account"
    ],
    "members": [
        [
            "Project Number",
            "Request Date",
            "Project Type",
            "Project Investment"
        ]
    ]
}
]
}
}

```

OR

No dimension names provided is less efficient:

```

{
  "exportPlanningData": true,
  "gridDefinition": {
    "suppressMissingBlocks": true,
    "pov": {
      "members": [
        [ "BaseData" ], [ "FY15" ], [ "Plan" ], [ "Working" ], [ "410" ],
["P_160" ]
      ]
    },
    "columns": [
      {
        "members": [
          [ "IDescendants(Q1)" ]
        ]
      },
      {
        "members": [
          [ "IDescendants(Q2)" ]
        ]
      }
    ],
    "rows": [
      {
        "members": [
          [
            "Project Number",
            "Request Date",
            "Project Type",
            "Project Investment"
          ]
        ]
      }
    ]
  }
}

```

```

    ]
  }
]
}
}

```

JSON Output

The following shows an example of the response body with `exportPlanningData : true`.

```

{
  "pov": [
    "BaseData",
    "FY15",
    "Plan",
    "Working",
    "410",
    "P_160"
  ],
  "columns": [
    [
      "Jan",
      "Feb",
      "Mar",
      "Q1",
      "Apr",
      "May",
      "Jun",
      "Q2"
    ]
  ],
  "rows": [
    {
      "headers": [
        "Project Number"
      ],
      "data": [
        "1",
        "2",
        "3",
        " ",
        " ",
        " ",
        " ",
        " "
      ],
      "cellNotes": [
        [
          {
            "contents": "Internal Project<br/>"
          },
          {

```

```
        "contents": "Project delayed<br/>"
      }
    ],
    [],
    [],
    [],
    [],
    [],
    [],
    []
  ]
},
{
  "headers": [
    "Request Date"
  ],
  "data": [
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    ""
  ]
},
{
  "headers": [
    "Project Type"
  ],
  "data": [
    "Other",
    "IT",
    "Construction",
    "",
    "",
    "",
    "",
    ""
  ]
},
{
  "headers": [
    "Project Investment"
  ],
  "data": [
    "100000",
    "110000",
    "200000",
    "410000",
    "",
    "",
    "",
    ""
  ]
}
```

```

    ],
    "cellNotes": [
      [],
      [
        {
          "contents": "Internal + External investments made
here.<br/>"
        }
      ],
      [],
      [],
      [],
      [],
      []
    ],
    "supportingDetail": [
      null,
      {
        "items": [
          {
            "value": "60000",
            "position": 0,
            "label": "Internal",
            "generation": 0,
            "operator": "+"
          },
          {
            "value": "50000",
            "position": 1,
            "label": "External",
            "generation": 0,
            "operator": "+"
          }
        ]
      },
      null,
      null,
      null,
      null,
      null,
      null
    ]
  }
]
}

```

Example 2: Suppress missing blocks, rows, and columns when exporting a data slice:

```

{
  "exportPlanningData": false,

```

```
"gridDefinition": {
  "suppressMissingBlocks": true,
  "suppressMissingRows": true,
  "suppressMissingColumns": true,
  "pov": {
    "dimensions": [
      "HSP_View",
      "Year",
      "Scenario",
      "Version",
      "Entity",
      "Product"
    ],
    "members": [
      [
        "BaseData"
      ],
      [
        "FY15"
      ],
      [
        "Plan"
      ],
      [
        "Working"
      ],
      [
        "410"
      ],
      [
        "P_160"
      ]
    ]
  },
  "columns": [
    {
      "dimensions": [
        "Period"
      ],
      "members": [
        [
          "IDescendants(Q1)"
        ]
      ]
    },
    {
      "dimensions": [
        "Period"
      ],
      "members": [
        [
          "IDescendants(Q2)"
        ]
      ]
    }
  ]
}
```

```
],
  "rows": [
    {
      "dimensions": [
        "Account"
      ],
      "members": [
        [
          "Project Number",
          "Request Date",
          "Project Type",
          "Project Investment"
        ]
      ]
    }
  ]
}
```

JSON Output:

```
{
  "pov": [
    "BaseData",
    "FY15",
    "Plan",
    "Working",
    "410",
    "P_160"
  ],
  "columns": [
    [
      "Jan",
      "Feb",
      "Mar",
      "Q1",
      "Apr",
      "May",
      "Jun",
      "Q2"
    ]
  ],
  "rows": [
    {
      "headers": [
        "Project Number"
      ],
      "data": [
        "1",
        "2",
        "3",

```

```

        " "
    ]
},
{
    "headers": [
        "Project Type"
    ],
    "data": [
        "Other",
        "IT",
        "Construction",
        ""
    ]
},
{
    "headers": [
        "Project Investment"
    ],
    "data": [
        "100000",
        "110000",
        "200000",
        "410000"
    ]
}
]
}

```

Example payload with multiple dimensions:

```

{
    "exportPlanningData": false,
    "gridDefinition": {
        "suppressMissingBlocks": true,
        "pov": {
            "dimensions": [
                "HSP_View",
                "Scenario",
                "Version",
                "Product"
            ],
            "members": [
                [
                    "BaseData"
                ],
                [
                    "Plan"
                ],
                [
                    "Working"
                ],
                [

```



```
        "P_160"
      ]
    ],
    "columns": [
      {
        "dimensions": [
          "Year", "Period"
        ],
        "members": [
          [
            "FY19"
          ],
          [
            "IDescendants(Q1)"
          ]
        ]
      },
      {
        "dimensions": [
          "Year", "Period"
        ],
        "members": [
          [
            "FY20"
          ],
          [
            "IDescendants(Q1)"
          ]
        ]
      }
    ],
    "rows": [
      {
        "dimensions": [
          "Entity", "Account"
        ],
        "members": [
          [
            "410",
            "420"
          ],
          [
            "Project Number",
            "Request Date",
            "Project Type",
            "Project Investment"
          ]
        ]
      },
      {
        "dimensions": [
          "Entity", "Account"
        ],
        "members": [
```

```

    [
      "430"
    ],
    [
      "Project Investment"
    ]
  ]
}
]
}
}
}

```

Clear Data Slices

Can be used to clear Planning and Essbase data for a specified region. In order to run this operation, the user must be an administrator.

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/cleardataslice

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-72 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to export the data slice	Path	Yes	None
plantype	Name of the plan type for which to export the data slice	Path	Yes	None

Example URL and Payload:

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/Vision/plantypes/plan1/cleardataslice

Payload Parameters

The Payload is JSON with the following parameters.

Table 7-73 Parameters

Name	Description
gridDefinition	JSON grid definition to define the region
clearEssbaseData	True or false. If true, will clear Essbase numeric data. Default is true.
clearPlanningData	True or false. If true, will delete the cell notes, attachments, and supporting details. Default is false.

Sample Payload

Providing dimension names as follows in the `gridDefinition` is recommended and is more efficient.

```
{
  "clearEssbaseData":true,
  "clearPlanningData":false,
  "gridDefinition" : {
    "suppressMissingBlocks" : true,
    "pov" : {
      "dimensions" : [ "HSP_View", "Year", "Scenario", "Version", "Entity", "Product" ],
      "members" : [ [ "BaseData" ], [ "FY15" ], [ "Plan" ], [ "Working" ], [ "410" ], [ "P_160" ] ]
    },
    "columns" : [ {
      "dimensions" : [ "Period" ],
      "members" : [ [ "IDescendants(Q1)" ] ]
    }, {
      "dimensions" : [ "Period" ],
      "members" : [ [ "IDescendants(Q2)" ] ]
    } ],
    "rows" : [ {
      "dimensions" : [ "Account" ],
      "members" : [ [ "Project Number", "Request Date", "Project Type", "Project Investment" ] ]
    } ]
  }
}
```

OR

No dimension names provided is less efficient:

```
{
  "clearEssbaseData":true,
  "clearPlanningData":false,
  "gridDefinition" : {
    "suppressMissingBlocks" : true,
    "pov" : {
      "members" : [ [ "BaseData" ], [ "FY15" ], [ "Plan" ], [ "Working" ], [ "410" ], [ "P_160" ] ]
    },
    "columns" : [ {
      "members" : [ [ "IDescendants(Q1)" ] ]
    }, {
      "members" : [ [ "IDescendants(Q2)" ] ]
    } ],
    "rows" : [ {
      "members" : [ [ "Project Number", "Request Date", "Project Type", "Project Investment" ] ]
    } ]
  }
}
```

Response

Supported Media Types: `application/json`

JSON Output

The following shows an example of the response body with `clearEssbaseData` true and `clearPlanningData` false. There is one rejected cell due to the presence of supporting details because `clearPlanningData` is false:

```
{
  "numClearedCells": 31,
  "numRejectedCells": 1
  "rejectedCells": ["Project Investment,Feb,BaseData,FY15,Plan,Working,410,P_160"],
}
```

Getting and Setting Substitution Variables for Planning

You can get and set substitution variables at the plan level and application level using a set of REST resources, as summarized here.

Required Roles

Service Administrator, Power User (with Rule Launch access)

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 7-74 Getting and Setting Substitution Variables

Task	Request	REST Resource
Get All Substitution Variables Defined for the Application	GET	/HyperionPlanning/rest/ {api_version}/applications/ {application}/ substitutionvariables
Get a Substitution Variable Defined for the Application	GET	/HyperionPlanning/rest/ {api_version}/applications/ {application}/ substitutionvariables/MyPeriod
Create or Update All Substitution Variables Defined for the Application	POST	/HyperionPlanning/rest/ {api_version}/applications/ {application}/ substitutionvariables
Get Substitution Variables Defined at the Plan Type Level	GET	/HyperionPlanning/rest/ {api_version}/applications/ {application}/plantypes/ {plantype}/ substitutionvariables
Get Derived Substitution Variables at the Plan Type Level	GET	/HyperionPlanning/rest/ {api_version}/applications/ {application}/plantypes/ {plantype}/ substitutionvariables? q={"derivedValues":true}
Get a Substitution Variable Defined at the Plan Type Level	GET	/HyperionPlanning/rest/ {api_version}/applications/ {application}/plantypes/ {plantype}/ substitutionvariables/CurrYear
Get a Derived Substitution Variable Defined at the Plan Type Level	GET	HyperionPlanning/rest/ {api_version}/applications/ {application}/plantypes/ {plantype}/ substitutionvariables/MyPeriod? q={"derivedValues":true}
Create and Update Substitution Variables at the Plan Type Level	POST	/HyperionPlanning/rest/ {api_version}/applications/ {application}/plantypes/ {plantype}/ substitutionvariables

Get All Substitution Variables Defined for the Application

Can be used to retrieve all substitution variables defined for the application (all plan types).

Required Roles

Service Administrator, Power User (with Rule Launch access)

REST Resource

GET /HyperionPlanning/rest/{api_version}/applications/{application}/substitutionvariables

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-75 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/Vision/substitutionvariables

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-76 Parameters

Name	Description
items	Collection of information about the resource
name	Name of the substitution variable, such as CurrYear
value	Value of the substitution variable, such as FY16
planType	Plan type, such as Plan1, or ALL for all plan types

Example of Response Body

The following shows an example of the response body.

```
{
  "items": [{
    "name": "CurrYear",
    "value": "FY16",
    "planType": "ALL"
  }, {
```

```

        "name": "CurrYear",
        "value": "FY17",
        "planType": "Plan2"
    }, {
        "name": "CurrPeriod",
        "value": "Jan",
        "planType": "Plan1"
    }, {
        "name": "CurrPeriod",
        "value": "Feb",
        "planType": "ALL"
    }
  ]
}

```

Get a Substitution Variable Defined for the Application

Can be used to retrieve a substitution variable defined for the application.

Required Roles

Service Administrator, Power User (with Rule Launch access)

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/
{application}/substitutionvariables/CurrPeriod
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-77 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/substitutionvariables/
CurrPeriod
```

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-78 Parameters

Name	Description
name	Name of the substitution variable, such as CurrPeriod
value	Value of the substitution variable, such as Jan
planType	Plan type, such as Plan1, or ALL for all plan types

Example of Response Body

The following shows an example of the response body.

```
{
  "name": "CurrPeriod",
  "value": "Jan",
  "planType": "ALL",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/vision/substitutionvariables/CurrPeriod",
    "action": "GET"
  }]
}
```

Create or Update All Substitution Variables Defined for the Application

Can be used to create or update substitution variables for the application.

Variables in the payload that exist in the application at the defined scope will be updated; new variables will be created at the defined scope.

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/substitutionvariables

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-79 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None

Table 7-79 (Cont.) Parameters

Name	Description	Type	Required	Default
application	The name of the application	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/substitutionvariables
```

Example Payload

With the following payload, CurrYear at the application level will be updated, and CurrPeriod will be created at the Plan3 level.

```
{
  "items": [{
    "name": "CurrYear",
    "value": "FY18",
    "planType": "ALL"
  }, {
    "name": "CurrPeriod",
    "value": "Dec",
    "planType": "Plan3"
  }]
}
```

Response

Supported Media Types: application/json

Example of a successful response

Http status code: 204 (No content)

Example of an error response

Http status: 400

To confirm the results, you can go to the application to see the updates.

Get Substitution Variables Defined at the Plan Type Level

Can be used to retrieve a list of retrieve a list of substitution variables defined at the plan type level.

Required Roles

Service Administrator, Power User (with Rule Launch access)

Rest Resource

GET /HyperionPlanning/rest/{api_version}/applications/{application}/plantypes/{plantype}/substitutionvariables

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-80 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
plantype	The plan type for which to get substitution variables	Path	Yes	None

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/Vision/plantypes/Plan1/substitutionvariables

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-81 Parameters

Name	Description
items	Collection of information for the resource
name	Name of the substitution variable, such as CurrPeriod
value	Value of the substitution variable, such as Jan
planType	Name of the plan type, such as Plan1

Example of Response Body

The following shows an example of the response body.

```
{
  "items": [{
    "name": "CurrPeriod",
    "value": "Jan",
    "planType": "Plan1"
  }]
}
```

}

Get Derived Substitution Variables at the Plan Type Level

Can be used to retrieve a list of derived substitution variables at the plan type level.

Required Roles

Service Administrator, Power User (with Rule Launch access)

Rest Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/
{application}/plantypes/{plan}/substitutionvariables?
q={"derivedValues":true}
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-82 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
plantype	Name of the plan type for which to get substitution variables	Path	Yes	None
q	Options for returning derived values, as described in the following row	Query	No	
derivedValues	If set to true, the derived list shows all variables available at run time to be used against the plan type. This includes variables defined at the application level and at the plan type level. If a substitution variable with the same name is defined both at the plan type level and at the application level, the plan type level is returned.	Query	No	derivedValues = true when getting derived substitution variables

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/plantypes/Plan1/
substitutionvariables?q={"derivedValues":true}
```

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-83 Parameters

Name	Description
items	Collection of information about the resource
name	Name of the substitution variable, such as CurrYear
value	Value of the substitution variable, such as FY16
planType	Name of the plan type, such as Plan1, or ALL for all plan types

Example of Response Body

The following shows an example of the response body.

```
{
  "items": [{
    "name": "CurrYear",
    "value": "FY16",
    "planType": "ALL"
  }, {
    "name": "CurrPeriod",
    "value": "Jan",
    "planType": "Plan1"
  }]
}
```

Get a Substitution Variable Defined at the Plan Type Level

Can be used to retrieve a substitution variable defined at the plan type level.

Required Roles

Service Administrator, Power User (with Rule Launch access)

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/
plantypes/{plantype}/substitutionvariables/CurrYear
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-84 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/vision/plantypes/plan1/
substitutionvariables/CurrYear
```

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-85 Parameters

Name	Description
name	Name of the substitution variable, such as CurrYear
value	Value of the substitution variable, such as FY16
planType	Name of the plan type, such as Plan1

Example of Response Body

The following shows an example of the response body.

```
{
  "name": "CurrYear",
  "value": "FY17",
  "planType": "Plan1",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/vision/plantypes/plan1/substitutionvariables/
CurrYear",
    "action": "GET"
  }]
}
```

Get a Derived Substitution Variable Defined at the Plan Type Level

Can be used to retrieve a derived substitution variable defined at the plan type level.

Required Roles

Service Administrator, Power User (with Rule Launch access)

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/
{application}/plantypes/{plan}/substitutionvariables/CurrPeriod?
q={"derivedValues":true}
```

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-86 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
plan	Name of the plan type for which to get substitution variables	Path	Yes	None
q	Options for returning derived values. The value is described in the following row.	Query	No	derivedValues = false
derivedValues	true If set to true, the derived list shows all variables available at run time to be used against the plan type. This includes variables defined at the application level and at the plan type level. If a substitution variable with the same name is defined both at the plan type level and at the application level, the plan type level is returned.	Query	No	false

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/v3/applications/Vision/plantypes/Plan1/  
substitutionvariables?q={"derivedValues"=true}
```

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 7-87 Parameters

Name	Description
items	Collection of information about the resource
name	Name of the substitution variable, such as CurrYear
value	Value of the substitution variable, such as FY16
planType	Name of the plan type, such as Plan1, or ALL for all plan types

Example of Response Body

The following shows an example of the response body.

```

{
  "name": "CurrPeriod",
  "value": "Jan",
  "planType": "ALL",
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/vison/plantypes/plan1/substitutionvariables/
CurrPeriod?q=%7B%22derivedValues%22:true%7D",
    "action": "GET"
  }]
}

```

Create and Update Substitution Variables at the Plan Type Level

Can be used to create and update substitution variables at the plan type level. Variables in the payload that exist at the plan type level are updated. New variables are created at the plan type level.

Required Roles

Service Administrator

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/
{application}/plantypes/{plantype}/substitutionvariables

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 7-88 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
plantype	Name of the plan type for which to get and set substitution variables	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/Vision/plantypes/Plan1/
substitutionvariables
```

Response

Supported Media Types: application/json

Example of a successful response

Http status code: 204 (No content)

Example of an error response

Http status: 400

To confirm the results, you can go to the application to see the updates.

Working with Connections

Use these REST APIs to work with connections.

With multiple environments, using REST APIs saves you time and effort by automating the process of logging in and configuring connections. For information about accessing environments, see [Accessing EPM Cloud](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using these REST APIs requires prerequisites. See [Prerequisites](#).

Table 7-89 Working with Connections

Task	Request	REST Resource
View a Connection	GET	<i>/HyperionPlanning/rest/epm/{api_version}/applications/{ application }/connections/{connectionRef}</i>
View all Connections	GET	<i>/HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections</i>
Update a Connection	POST	<i>/HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections/{connectionRef}</i>

View a Connection

Use this REST API to view details for a connection that is saved in an application.

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections/{connectionRef}

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-90 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to view the connection	Path	Yes	None
connectionRef	The connection to view. The value can be either a connection name or ID.	Path	Yes	None

Response

Supported Media Types: application/json

Payload Parameters:

The following table summarizes the parameters.

Table 7-91 Parameters

Parameters	Description
items	Collection of information about the resource
id	Unique identifier for the connection, such as 1c89922d-92ba-46c1-850f-e2a8a416ddf2
name	Name of the connection, such as Connection29
description	Description of the connection, such as Planning
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning
username	The username for the connection, such as admin
modified	The time stamp of the last modification to the connection details, such as 2021-01-18 12:23:49.0
modifiedBy	The last service administrator to modify the connection details, such as admin

Example Response

```
{
  "id": "f83b3da2-9505-415e-b7f7-3cf113cc94e4",
  "name": "Connection1",
  "description": "Test Connection",
```

```

    "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
    "username": "admin",
    "modified": "2021-02-02 09:16:02.0",
    "modifiedBy": "admin",
    "links": [
      {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections/f83b3da2-9505-415e-
b7f7-3cf113cc94e4",
        "action": "GET",
        "rel": "self",
        "data": null
      }
    ]
  }
}

```

View All Connections

Use this REST API to view details for all of the connections saved in an application.

This API supports paging, so you can filter the number of connections you see in the output using the offset and limit parameters shown in the table.

Required Roles

Service Administrator

REST Resource

GET */HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections*

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-92 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to view connections	Path	Yes	None
offset	For paging of jobs. Indicates the actual index from which the records are returned. It is 0 based.	Query	No	0
limit	For paging for jobs. Controls how many items to return. Defaults to 25 if not specified.	Query	No	25

Example URL

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections?offset=2&limit=2
```

Response

The following table summarizes the parameters.

Table 7-93 Parameters

Parameters	Description
items	Collection of information about the resource
id	Unique identifier for the connection, such as 1c89922d-92ba-46c1-850f-e2a8a416ddf2
name	Name of the connection, such as Connection29
description	Description of the connection, such as Planning
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning
username	The username for the connection, such as admin
modified	The time stamp of the last modification to the connection details, such as 2021-01-18 12:23:49.0
modifiedBy	The last service administrator to modify the connection details, such as admin

Example Response

```
{
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections?offset=2&limit=2",
      "action": "GET",
      "rel": "self",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections?offset=0&limit=2",
      "action": "GET",
      "rel": "prev",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections?offset=4&limit=2",
      "action": "GET",
      "rel": "next",
      "data": null
    }
  ]
}
```

```

    ],
    "items": [
      {
        "id": "1c89922d-92ba-46c1-850f-e2a8a416ddf2",
        "name": "Connection20",
        "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
        "username": "admin",
        "modified": "2021-01-18 12:23:49.0",
        "modifiedBy": "admin",
        "links": [
          {
            "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections/1c89922d-92ba-46c1-850f-
e2a8a416ddf2",
            "action": "GET",
            "rel": "Self",
            "data": null
          }
        ]
      },
      {
        "id": "ec94a10e-717b-449a-89ce-0c16b1688caa",
        "name": "Connection29",
        "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
        "username": "admin",
        "modified": "2021-01-18 12:23:49.0",
        "modifiedBy": "admin",
        "links": [
          {
            "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections/
ec94a10e-717b-449a-89ce-0c16b1688caa",
            "action": "GET",
            "rel": "Self",
            "data": null
          }
        ]
      }
    ]
  },
  "type": null
}

```

Update a Connection

Use this REST API to update a specific connection that is saved in an application.

You can update the values using either a plain text password or encrypted password. The response returns the updated connection details.

Required Roles

Service Administrator

REST Resource

GET /HyperionPlanning/rest/epm/{api_version}/applications/{application}/connections/{connectionRef}

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 7-94 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application for which to update a connection	Path	Yes	None
connectionRef	The connection to refer to. The value can be either a connection name or ID.	Path	Yes	None

Table 7-95 Parameters for Connection Information that Can Be Modified

Name	Description
name	Name of the connection, such as Connection29
description	Description of the connection, such as Planning
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning
username	The username for the connection, such as admin
password	The password of the connection
encryptedPassword	The password of the connection in the encrypted format using the EPM Automate encrypt command. See encrypt .

Example Body

Example 1:

```
{
  "name": "<NEW_CONNECTION_NAME>",
  "description": "<NEW_DESCRIPTION>",
  "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
  "username": "<NEW_USERNAME>",
  "password": "<NEW_PASSWORD>"
}
```

Example 2:

```
{
  "name": "<NEW_CONNECTION_NAME>",
  "description": "<NEW_DESCRIPTION>",
  "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
  "username": "<NEW_USERNAME>",
  "encryptedPassword": "<ENCRYPTED_PASSWORD>"
}
```

Response

Supported Media Type: application/json

Table 7-96 Parameters

Parameters	Description
items	Collection of information about the resource
id	Unique identifier for the connection, such as 1c89922d-92ba-46c1-850f-e2a8a416ddf2
name	Name of the connection, such as Connection29
description	Description of the connection, such as Planning
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning
username	The username for the connection, such as admin
modified	The time stamp of the last modification to the connection details, such as 2021-01-18 12:23:49.0
modifiedBy	The last service administrator to modify the connection details, such as admin

```
{
  "id": "<ID>",
  "name": "<NEW_CONNECTION_NAME>",
  "description": "<NEW_DESCRIPTION>",
  "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning",
  "username": "<NEW_USERNAME>",
  "modified": "2021-02-02 09:16:02.0",
  "modifiedBy": "admin",
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/epm/v1/applications/epbcs1/connections/f83b3da2-9505-415e-
b7f7-3cf113cc94e4",
      "action": "GET",
      "rel": "self",
      "data": null
    }
  ]
}
```

```
} ]
```

8

Migration REST APIs

Use the Migration REST APIs to get API versions, work with files, manage services and application snapshots, work with users, and skip updates.

Some Migration REST APIs are version 11.1.2.3.600 and others are version v1 or v2. Passing the incorrect version will result in 404 errors when the API is invoked. Be sure to use the correct version for the API. Be sure to use the correct version for the API. Migration REST API versions are as follows.

These Migration APIs are version 11.1.2.3.600:

- [Get Information About All Services](#)
- [Add Users To a Group](#)
- [Remove Users from a Group](#)
- [Run Recreate on a Service \(11.1.2.3.600\)](#)
- [Get Information About All Application Snapshots](#)
- [Get Information About a Specific Application Snapshot](#)
- [Provide Feedback \(v11.1.2.3.600\)](#)
- [LCM Export \(v1\)](#)
- [LCM Import \(v1\)](#)
- [Upload](#)
- [Download](#)
- [List Files \(v11.1.2.3.600\)](#)
- [Delete Files \(v11.1.2.3.600\)](#)

These Migration APIs are version v1:

- [Copy Application Snapshot \(v1\)](#)
- [Upload Application Snapshot](#)
- [Download Application Snapshot](#)
- [Rename Application Snapshot \(v1\)](#)
- [Copy to Object Store \(v1\)](#)
- [Copy from Object Store \(v1\)](#)
- [Restart the Service Instance \(v1\)](#)
- [Get the Build Version and Daily Maintenance Window Time \(v1\)](#)
- [Setting the Daily Maintenance Time \(v1\)](#)
- [Running Daily Maintenance While Skipping the Scheduled Daily Maintenance \(v1\)](#)
- [Adding users to an Identity Domain](#)
- [Removing users from an identity domain](#)

- Adding users to a predefined role or application role
- Removing users' role assignment
- Updating users
- Add Users to a Group
- Remove Users from a Group
- Adding a user to a batch of groups
- Removing a user from a batch of groups
- Adding groups
- Removing groups
- Role assignment report
- Update Users
- User Group Report
- User Audit Report (v1)
- User Access Report (v1)
- Role Assignment Report
- Role Assignment Audit Report for OCI (Gen 2) Environments
- Invalid Login Report for OCI (Gen 2) Environments
- Set Encryption Key (v1)
- Manage Permission for Manual Access to Database (v1)
- Copy a File Between Instances (v1)
- Skip Updates (v1)
- Send Email (v1)
- Clone an Environment

These Migration APIs are version v2:

- Copy Application Snapshot (v2)
- Copy to Object Store (v2)
- Copy from Object Store (v2)
- Restart the Service Instance (v2)
- Run Recreate on a Service (v2)
- Get the Build Version and Daily Maintenance Window Time (v2)
- Setting the Daily Maintenance Time (v2)
- Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)
- List Backups - Only for OCI (Gen 2) Environments (v2)
- Restore Backup - Only for OCI (Gen 2) Environments (v2)
- View the IP Allowlist - Only for OCI (Gen 2) Environments
- Update the IP Allowlist - Only for OCI (Gen 2) Environments
- Group Assignment Audit Report

- [Set Encryption Key \(v2\)](#)
- [Manage Permission for Manual Access to Database \(v2\)](#)
- [Copy a File Between Instances \(v2\)](#)
- [List Files \(v2\)](#)
- [Delete Files \(v2\)](#)
- [Skip Updates \(v2\)](#)
- [Send Email \(v2\)](#)
- [LCM Import \(v2\)](#)
- [LCM Export \(v2\)](#)
- [Provide Feedback \(v2\)](#)
- [Rename Application Snapshot \(v2\)](#)
- [User Access Report \(v2\)](#)
- [User Audit Report \(v2\)](#)
- [Export Essbase Data \(v2\)](#)

This Migration API is version v3:

[Delete Files \(v3\)](#)

URL Structure for Migration

This topic shows the general URL structure for the Migration REST APIs.

Some Migration REST APIs are version 11.1.2.3.600 and others are version v1 or v2. Be sure to use the correct version for the API. Passing the incorrect version will result in 404 errors when the API is invoked.

You can find an API version using REST APIs as described here: [Getting API Versions for Migration APIs](#). For a list of all of the Migration APIs and their version numbers, see [Migration REST APIs](#).

Use this URL structure to access the Migration REST resources:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
interop/rest/{api_version}/{path}
```

Where:

api_version—the API version you are developing with

path—Defines the resource

Getting API Versions for Migration APIs

You can manage versions using the set of REST resources summarized in the following table.

Important: The version number is case-sensitive. For example, if the version number is listed as v1 with a lowercase v, you cannot enter the version number with a capital V as in this

incorrect example, v1, which would result in an error. Instead, you must enter the version number with a lowercase v as in this correct example: v1.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-1 Manage Versions of Migration APIs

Task	Request	REST Resource
Get REST API Versions for Migration	GET	/interop/rest/
Get Information About a Specific REST API Version for Migration	GET	/interop/rest/{apiVersion}

Get REST API Versions for Migration

Returns information about which REST APIs are available and supported. Multiple versions may be supported simultaneously.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /interop/rest/

Request

Supported Media Types: application/json

Parameters:

Table 8-2 Parameters

Name	Description
Details	In case of errors, details are published with the error string
Status	See Migration Status Codes
Items	Detailed information about the API
Version	The version
Lifecycle	Possible values: active, deprecated
Latest	Whether this version is the latest
Links	Detailed information about the link
Href	Links to API call
Action	The HTTP call type
Rel	Possible values: self
Data	Parameters as key value pairs passed in the request
lifecycle	The stage in the lifecycle, such as active
version	The version, such as 11.1.2.3.600, v1, and v2, for example, "version": "11.1.2.3.600", "v1", v2"

Table 8-2 (Cont.) Parameters

Name	Description
serviceType	The service type, such as PCMCS
serverVersion	The server version, such as 21.05.70
buildVersion	The build version, such as 21.05.52

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [
    {
      "href": "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600",
      "rel": "self",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": 0,
  "items": [
    {
      "latest": true,
      "links": [
        {
          "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600",
          "rel": "version",
          "data": null,
          "action": "GET"
        }
      ],
      "lifecycle": "active",
      "version": "11.1.2.3.600",
      "serviceType": "PCMCS",
      "serverVersion": "21.05.70",
      "buildVersion": "21.05.52"
    }
  ],
}
```

Getting API Versions of Migration APIs Sample Code

Prerequisites: json.jar

Common functions: See [Common Helper Functions for Java](#)

Example 8-1 Java Sample – getVersionsOfLCM.java

```
//
//
// BEGIN - List all the versions in PBCS
//
public void getLCMVersions() throws Exception {
    String urlString = String.format("%s/interop/rest", serverUrl);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        JSONArray fileList = json.getJSONArray("items");
        System.out.println("List of files are :");
        JSONObject jsonObj = null;
        for(int i=0; i<fileList.length();i++){
            System.out.println("Version :" + jsonObj.getString("version"));
            System.out.println("Lifecycle :" +
jObj.getString("lifecycle"));
            System.out.println("Latest :" + jsonObj.getString("latest"));
            System.out.println("Link :" + ((JSONObject) ((JSONArray)
jObj.getJSONArray("links")).get(0)).getString("href") + "\n");
        }
    }
}
//
// END - List all the versions in PBCS
//
```

Example 8-2 cURL Sample – GetVersionsOfLCM.sh

Prerequisites: jq

Common functions: See [Common Helper Functions for cURL](#)

```
funcGetLCMVersions() {
    url=$SERVER_URL/interop/rest
    funcExecuteRequest "GET" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "List of versions :"
        count=`echo $output | jq '.items | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Version : " `echo $output | jq
'.items['$i'].version'`
            echo "Lifecycle :" `echo $output | jq
'.items['$i'].lifecycle'`
            echo "Latest :" `echo $output | jq '.items['$i'].latest'`
            echo "Link :" `echo $output | jq
'.items['$i'].links[0].href'`
            echo ""
            i=`expr $i + 1`
        done
    fi
}
```

```

        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 8-3 Groovy Sample – GetVersionsOfLCM.groovy

Prerequisites: json.jar

Common functions: See [CSS Common Helper Functions for Groovy](#)

```

def getLCMVersions() {
    def url;
    try {
        url = new URL(serverUrl + "/interop/rest/")
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "GET", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        def items = object.items
        println "List of versions :"
        items.each{
            println "Version : " + it.version
            println "Lifecycle : " + it.lifecycle
            println "Latest : " + it.latest
            println "Link : " + it.links[0].href + "\n"
        }
    } else {
        println "Error occurred while listing versions"
        if (object.details != null)
            println "Error details: " + object.details
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Get Information About a Specific REST API Version for Migration

Returns information about a specific version.

Some Migration REST APIs are version 11.1.2.3.600, and others are other versions, such as v1 or v3. Passing the incorrect version will result in 404 errors when the API is invoked. Be sure to use the correct version for the API; `api_version` could be 11.1.2.3.600, v1, or v2.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /interop/rest/{api_version}

Request

Parameters:

The following table summarizes the client request.

Table 8-3 Parameters

Name	Description	Type	Default
api_version	Specific API version	Path	Yes

Response

Supported Media Types: application/json

Table 8-4 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values: self, recreate service
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status":0,
  "details":null,
  "links":[{"
    "data":null,
    "action":"GET",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600",
    "rel":"self"
  },{
    "data":null,
    "action":"GET",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
```

```

11.1.2.3.600/services",
    "rel":"recreate service"
  },{
    "data":null,
    "action":"GET",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applications",
    "rel":"application service"
  },{
    "data":null,
    "action":"GET",    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots    "rel":"application snapshot service"
  },{
    "data":null,
    "action":"POST",
    "rel":"feedback services",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
feedback"
  ]
}

```

Get Information about a Specific Version of Migration Sample Code

Example 8-4 Java Sample – getInfoAboutSpecificVersion.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```

//
// BEGIN - List version details
//
public void getLCMVersionDetails() throws Exception {
    String urlString = String.format("%s/interop/rest/%s", serverUrl,
apiVersion);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        JSONArray linksArray = json.getJSONArray("links");
        System.out.println("Version " + apiVersion + " details :");
        JSONObject jsonObj = null;
        for(int i=0; i < linksArray.length(); i++){
            jsonObj = (JSONObject)linksArray.get(i);
            System.out.println("Service :" + jsonObj.getString("rel"));
            System.out.println("URL :" + jsonObj.getString("href"));
            System.out.println("Action :" + jsonObj.getString("action") + "\n");
        }
    }
}
//

```



```
// END - List version details
//
```

Example 8-5 cURL Sample – GetInfoAboutSpecificVersion.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
funcGetLCMVersionDetails() {
    url=$SERVER_URL/interop/rest/$API_VERSION
    funcExecuteRequest "GET" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Version $API_VERSION details :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links[$i].rel'`
            echo "URL : " `echo $output | jq '.links[$i].href'`
            echo "Action : " `echo $output | jq '.links[$i].action'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Example 8-6 Groovy Sample – GetInfoAboutSpecificVersion.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def getLCMVersionDetails() {
    def url;
    try {
        url = new URL(serverUrl + "/interop/rest/" + apiVersion)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "GET", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        def links = object.links
        println "Version " + apiVersion + " details :"
        links.each{
            println "Service : " + it.rel
        }
    }
}
```

```

        println "URL : " + it.href
        println "Action : " + it.action + "\n"
    }
} else {
    println "Error occurred while fetching version details"
    if (object.details != null)
        println "Error details: " + object.details
}
}
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Upload and Download Files

Upload and download files.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-5 Upload and Download Files

Task	Request	REST Resource
Upload	POST	/interop/rest/11.1.2.3.600/applicationsnapshots/{applicationSnapshotName}/contents
Download	GET	/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents

Upload

Uploads a file from the current directory on the local machine to the repository. Files on the repository cannot be accessed directly.

If a file already exists, the API gives an error and does not overwrite it. Use this command to upload data, metadata, and back up snapshots to a service instance. See [About the EPM Automate Utility](#).

If a -1 status is returned and it is the last chunk to be uploaded, this means that an LCM artifact snapshot has been uploaded and zip extraction is in progress. The client pings the URL until the status is a positive integer. This job is done asynchronously.

Note: The entire path to the file must be encoded, for example, changing / to %5C and spaces to %20.

For example, change this path to an .HTML file in the `apr` directory:

```
apr/2020-03-04_23_07_20/2020-03-04_23_07_20.html
```

to this:

apr%5C2020-03-04%2023_07_20%5C2020-03-04%2023_07_20.html

This REST API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/11.1.2.3.600/applicationsnapshots/
{applicationSnapshotName}/contents

Note: For Data Management uploads, use the following JSON format for the query parameter:

```
/interop/rest/11.1.2.3.600/applicationsnapshots/{applicationSnapshotName}/
contents?extDirPath=inbox
```

Request

Supported Media Types: application/octet-stream

Parameters:

The following table summarizes the client request.

Table 8-6 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
applicationSnaps hotName	Name of the application snapshot or file name to be uploaded (for example, "Artifact Snapshot.zip" or s112.csv). A file with this name is created in the repository. If a file or folder with this name exists in the repository, an error indicates that a file or folder exists	Path	Yes	None

Table 8-6 (Cont.) Parameters

Name	Description	Type	Required	Default
extDirPath	<p>Used to support upload of Data Management files.</p> <p>Supported values include:</p> <ul style="list-style-type: none"> inbox - Upload files into the inbox; except for Profitability and Cost Management, EPM Cloud business processes look in this location for files to process profitinbox - Upload files to be processed by Profitability and Cost Management to_be_imported - Upload a Narrative Reporting snapshot that is to be imported during the next daily maintenance of the environment outbox - Upload files to the outbox for all EPM Cloud business processes except for Profitability and Cost Management profitoutbox - Upload files to the outbox for Profitability and Cost Management <p>You can also use a directory under inbox, outbox, profitinbox, and profitoutbox, for example, inbox/directory_name</p> <p>Example: "extDirPath= inbox/directory_name" to upload files to a directory within the inbox for processing by Data Management.</p> <p>Upload: /interop/rest/11.1.2.3.600/applicationsnapshots/applicationSnapshotName/contents?extDirPath=inbox%2Fdm_folder</p> <p>Example: "extDirPath=inbox" where inbox is the folder where the Data Management file is to be uploaded</p> <p>Example of query parameters in JSON format for Data Management Upload: /interop/rest/11.1.2.3.600/applicationsnapshots/{applicationSnapshotName}/contents?extDirPath=inbox</p>	Query	No	None

Response Body

Supported Media Types: application/json

Table 8-7 Parameters

Name	Description
details	In case of errors, details are published with the error string

Table 8-7 (Cont.) Parameters

Name	Description
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values: self, recreate service
data	Parameters as key value pairs passed in the request

Example of Response Body:

The following shows an example of the response body in JSON format.

```
{
  "status":0,
  "details":null,
  "links":[{"
    "data":null,
    "action":"POST",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/{applicationSnapshotName}/contents
```

REST API Examples with Postman

See [REST API Examples with Postman](#).

Upload Sample Code

Example 8-7 Java Sample – 11.1.2.3.600

Prerequisites: json.jar

```
/**
 *
 * Simple Implementation class to execute Upload functionality for API
 * version 11.1.2.3.600
 */
public class UploadFile
{

    private final static String userName ; // User name
    private final static String password ; // Password
    private final static String serverUrl; // Server URL
    private String filePath ; //zip File to be Uploaded
    private String extDirPath = "inbox"; // keep it null for uploading
    to root directory
    private String details = null;

    public void uploadFile() {
```

```
boolean status = true;

try {
    status = sendFileContents(filePath, extDirPath);

    if(status)
        System.out.println("Uploaded contents to " + new
File(filePath).getName());
    else
        System.err.println(details);

} catch (Exception e) {
    e.printStackTrace();
}
}

private boolean sendFileContents(String filePath, String extDirPath)
throws Exception {
    HttpURLConnection connection = null;
    FileInputStream content = null;
    File file = new File(filePath);

    try {
        String restURL = String.format(
            "%s/interop/rest/11.1.2.3.600/applicationsnapshots/%s/
contents",
            serverUrl, URLEncoder.encode(file.getName(), "UTF-8"));
        if(null != extDirPath)
            restURL = restURL + "?extDirPath="+extDirPath;
        URL url = new URL(restURL);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setInstanceFollowRedirects(false);
        connection.setDoOutput(true);
        connection.setUseCaches(false);
        connection.setDoInput(true);

        String creds = null;

        creds = userName + ":" + password;

        connection.setRequestProperty("Authorization",
            "Basic " + new
sun.misc.BASE64Encoder().encode(creds.getBytes()));
        connection.setRequestProperty("Content-Type", "application/octet-
stream");

        content = new FileInputStream(file);
        OutputStream paramOutputStream = connection.getOutputStream();
        if (content != null) {
            byte[] arrayOfByte = new byte[4096];
            boolean hasMore = true;
```

```
        while (hasMore) {
            int j = content.read(arrayOfByte);
            if (j < 0) {
                hasMore = false;
                continue;
            }
            paramOutputStream.write(arrayOfByte, 0,
j);
        }
    }

    int statusCode = connection.getResponseCode();

    String responseBody =
getStringFromInputStream(connection.getInputStream());
    if (statusCode == 200 && responseBody != null) {
        int commandStatus = getCommandStatus(responseBody);
        if (commandStatus == -1) {

getJobStatus(fetchPingUrlFromResponse(responseBody, "Job Status"),
"GET");
        }
        if (commandStatus == 0) {
            return true;
        }
        else{
            details = getDetails(responseBody);
        }
    }

    return false;
} finally {
    if(null != content)
        content.close();
    if (connection != null)
        connection.disconnect();
}
}

/**
 * Method to retrieve the error message
 * @param response
 * @return String details
 * @throws Exception
 */
private String getDetails(String response) throws Exception {
    JSONObject json = new JSONObject(response);
    if (!JSONObject.NULL.equals(json.get("details")))
        return json.getString("details");
    else
        return "NA";
}
```

```
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Download

Downloads a file from the repository to the current directory in the local environment.

If the content type of the response is `application/JSON`, then an error with details is displayed on the server. Otherwise, the content of the file is streamed through the response.

Note: The entire path to the file must be encoded, for example, changing `/` to `%5C` and spaces to `%20`.

For example, change this path to an `.HTML` file in the `apr` directory:

```
apr/2020-03-04_23_07_20/2020-03-04_23_07_20.html
```

to this:

```
apr%5C2020-03-04%2023_07_20%5C2020-03-04%2023_07_20.html
```

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

```
GET /interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/  
contents
```

Request

Parameters:

The following table summarizes the GET request parameters.

Table 8-8 Parameters

Name	Description	Type	Required	Default
applicationSnapshotName	<p>Application snapshot name or file name to download (for example, "Artifact Snapshot" or s112.csv).</p> <p>The entire applicationSnapshotName must be encoded before sending the request.</p> <p>To download a particular file, provide the path to that file as the value of applicationSnapshotName. For example, to download a Data Management file called s112.csv in the inbox, refer to the file as "inbox\s112.csv" in the path parameter.</p> <p>To download the Activity Reports or access log, use the fully qualified file name as shown in the output of List Files.</p> <p>For example, to download a specific file from the apr directory, use the following format:</p> <pre>pr%2F2020-03-04%0A23_07_20%2F2020-03-04%0A23_07_20.html apr%2F%0A2020-03-04%2023_07_20%2F%0Aaccess_log.zip apr%2F%0A2020-03-04%2023_07_20%2F%0Aactivityreport.json.</pre>	Path	Yes	None
api_version	Specific API version	Path	Yes	None
fileExtension	Specifies a file extension that can be used to create a file locally; can contain values such as zip and csv.	Path	Yes	None

Example of Request

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/Vision.zip/contents
```

Table 8-9 Parameters

Name	Description
Details	In case of errors, details are published with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to API call
Action	The HTTP call type
Rel	Possibly value: self

Table 8-9 (Cont.) Parameters

Name	Description
Data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format in case there is an error during download.

```
{
  "details": "Not a valid file.",
  "status": 8,
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/s112.csv/contents",
    "action": "GET",
    "rel": "self",
    "data": null
  }]
}
```

Download Sample Code**Example 8-8 Java Sample – downloadFile.java**

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#).

```
public class DownloadV1 {

    private String serverUrl ; // PBCS server URL
    private String apiVersion = "v1";
    private String userName ; // Server Username
    private String password ; //Server Password
    private static String fileName ; //snapshot to be downloaded.
    private String domain ;

    public void downloadFile(String fileName) throws Exception {
        HttpURLConnection connection = null;
        InputStream inputStream = null;
        FileOutputStream outputStream = null;

        try {
            fileName = fileName.replaceAll("/", "\\");
            URL url = new URL(
                String.format(
                    "%s/interop/rest/%s/applicationsnapshots/%s/
contents",
                    serverUrl, apiVersion,
```

```
        URLEncoder.encode(fileName, "UTF-8"));

    System.out.println("DOWNLOAD URL: " + url);
    connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("GET");
    connection.setInstanceFollowRedirects(false);
    connection.setDoOutput(true);
    connection.setUseCaches(false);
    connection.setDoInput(true);
    connection.setRequestProperty(
        "Authorization",
        "Basic "
            + new
sun.misc.BASE64Encoder().encode((userName
            + ":" + password).getBytes()));
    int status = connection.getResponseCode();
    if (status == 200) {
        if (connection.getContentType() != null
            && connection.getContentType().equals(
                "application/json")) {
            JSONObject json = new JSONObject(
                getStringFromInputStream(connection
                    .getInputStream()));
            System.out.println("Error downloading file : "
                + json.getString("details"));
        } else {
            String response =
getStringFromInputStream(connection
                .getInputStream());
            String pingURL = fetchPingUrlFromResponse(response,
                "Job Status");

            getJobStatusDownload(pingURL, "GET");
        }
    } else {
        throw new Exception("Http status code: " + status);
    }
} finally {
    if (connection != null)
        connection.disconnect();
    if (outputStream != null)
        outputStream.close();
    if (inputStream != null)
        inputStream.close();
}
}

private void downloadContent(String downloadURL) throws Exception {
    HttpURLConnection connection = null;
    InputStream inputStream = null;
    FileOutputStream outputStream = null;

    try {
        URL url = new URL(downloadURL);
```

```
connection = (URLConnection) url.openConnection();
connection.setRequestMethod("GET");
connection.setInstanceFollowRedirects(false);
connection.setDoOutput(true);
connection.setUseCaches(false);
connection.setDoInput(true);
connection.setRequestProperty(
    "Authorization",
    "Basic "
        + new sun.misc.BASE64Encoder().encode((userName
            + ":" + password).getBytes());
connection.setRequestProperty("Content-Type",
    "application/x-www-form-urlencoded");
int status = connection.getResponseCode();

if (status == 200) {
    if (connection.getContentType() != null
        && connection.getContentType().equals(
            "application/json")) {
        JSONObject json = new JSONObject(
            getStringFromInputStream(connection
                .getInputStream()));
        System.out.println("Error downloading file : "
            + json.getString("details"));
    } else {
        inputStream = connection.getInputStream();
        String downloadedFileName = fileName;
        if (fileName.lastIndexOf("/") != -1) {
            downloadedFileName = fileName.substring(fileName
                .lastIndexOf("/") + 1);
        }

        String ext = ".zip";
        if (connection.getHeaderField("fileExtension") != null) {
            ext = "." +
connection.getHeaderField("fileExtension");
        }
        if (fileName.lastIndexOf(".") != -1
            && fileName.lastIndexOf(".") != 0)
            ext = fileName.substring(fileName.lastIndexOf(".") +
1);

        outputStream = new FileOutputStream(new File(
            downloadedFileName + ext));
        int bytesRead = -1;
        byte[] buffer = new byte[5 * 1024 * 1024];
        while ((bytesRead = inputStream.read(buffer)) != -1)
            outputStream.write(buffer, 0, bytesRead);
        System.out.println("File download completed.");
    }
} else {
    throw new Exception("Http status code: " + status);
}
} finally {
```

```

        if (connection != null)
            connection.disconnect();
        if (outputStream != null)
            outputStream.close();
        if (inputStream != null)
            inputStream.close();
    }
}

private void getJobStatusDownload(String pingUrlString, String
methodType)
    throws Exception {
    boolean completed = false;
    while (!completed) {

        String pingResponse = executeRequest(pingUrlString,
methodType,
            null, "application/x-www-form-urlencoded");
        JSONObject json = new JSONObject(pingResponse);
        int status = json.getInt("status");
        if (status == -1) {
            try {
                System.out.println("Please wait...");
                Thread.sleep(20000);
            } catch (InterruptedException e) {
                completed = true;
                throw e;
            }
        } else {
            if (status > 0) {
                System.out.println("Error occurred: "
+ json.getString("details"));
            } else {
                String downloadURL =
fetchPingUrlFromResponse(pingResponse,
                    "Download link");
                downloadContent(downloadURL);
            }
            completed = true;
        }
    }
}
}
}

```

Example 8-9 cURL Sample – DownloadFile.shPrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [Common Helper Functions for cURL](#)

#!/bin/sh

SERVER_URL=""

```
USERNAME=""
PASSWORD="1"

API_VERSION="v1"
FILENAME=$1

funcDownloadContent(){
    output=`cat pingResponse.txt`
    count=`echo $output | jq '.links | length'`

    i=0
    pingUrlC=""
    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links['$i'].rel'`
        rel=`echo "$rel" | tr -d "\""`

        if [ "$rel" == "Download link" ]; then
            pingUrlC=`echo $output | jq
'.links['$i'].href'`
            pingUrlC=`echo "$pingUrlC" | tr -d "\""`
        fi
        i=`expr $i + 1`
    done

    #request has to be get
    statusWrite=`curl -s -w "%{http_code}" -u "$USERNAME:$PASSWORD" --
request GET -D "respHeader.txt" -o "$1" -H "Content-Type: application/x-www-
form-urlencoded" "$pingUrlC"`

    if [ $statusWrite == 200 ]; then

        contentType=`echo $(grep 'Content-Type:' respHeader.txt) |
tr -d [:space:]`
        #contentbody=`cat writeResponse.txt`
        if [ ! -z $contentType ] && [[ $contentType = *"application/
json"* ]]; then
            echo "Error occurred. "
        else
            fileExtension=`echo $(grep -r "fileExtension: "
respHeader.txt | awk '{print ($2)}') | tr -d [:space:]`

            if [ ! -z $fileExtension ]; then
                if [[ ! $filepath =~ \.$fileExtension$ ]]; then
                    mv "$1" "$1".$fileExtension
                fi
            fi
            echo "Downloade file successfully"
        fi
    fi
    funcRemoveTempFiles "response.txt" "respHeader.txt"
}

funcDownloadFile() {
```

```

        filepath="/u01/$FILENAME"
        encodedFileName=$(echo $FILENAME | sed -f urlencode.sed)
        url=$SERVER_URL/interop/rest/$API_VERSION/
        applicationsnapshots/$encodedFileName/contents

        statusCode=`curl -X GET -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -H "Content-Type: application/x-www-form-
urlencoded" -D "respHeader.txt" -o "response.txt" $url`

        if [ $statusCode == 200 ]; then

                contentType=`echo $(grep 'Content-Type:'
respHeader.txt) | tr -d [:space:]`
                contentbody=`cat response.txt`

                if [ -z $contentType ] && [[ $contentType =
*"application/json"* ]]; then
                        output=`cat $filepath`
                        error=`echo $output | jq '.details'`
                        echo "Error occurred. " $error
                        funcRemoveTempFiles $filepath
                else
                        funcGetStatus "GET"

                fi

                else

                        echo "Error executing request"
                        if [ $statusCode != 000 ]; then
                                echo "Response error code :

"$statusCode

                                funcPrintErrorDetails "response.txt"
                                funcRemoveTempFiles "respHeader.txt"

                                "response.txt"

                                fi
                                exit 0
                        fi

                fi

        #funcRemoveTempFiles "respHeader.txt" "response.txt"

    }
    funcDownloadFile $FILENAME

```

Example 8-10 Groovy Sample – DownloadFile.groovy

```

class DownloadV1 {

    def serverUrl ; // PBCS server URL
    def apiVersion = "v1";
    def userName ; //Server Username
    def password ; //Server Password
    def fileName ; //Snapshot to be downloaded

```

```
void downloadFile(def fileName) throws Exception {
    HttpURLConnection connection = null;
    InputStream inputStream = null;
    FileOutputStream outputStream = null;

    try {
        fileName = fileName.replaceAll("/", "\\");
        URL url = new URL(String.format("%s/interop/rest/%s/
applicationsnapshots/%s/contents", serverUrl,
        apiVersion, URLEncoder.encode(fileName, "UTF-8")));

        println "DOWNLOAD URL: "+url
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        connection.setInstanceFollowRedirects(false);
        connection.setDoOutput(true);
        connection.setUseCaches(false);
        connection.setDoInput(true);
        connection.setRequestProperty("Authorization",
            "Basic " + new sun.misc.BASE64Encoder().encode((userName
+ ":" + password).getBytes()));
        connection.setRequestProperty("Content-Type", "application/x-www-
form-urlencoded");
        int status = connection.getResponseCode();
        if (status == 200) {
            if (connection.getContentType() != null &&
connection.getContentType().equals("application/json")) {
                JSONObject json = new
JSONObject(getStringFromInputStream(connection.getInputStream()));
                println "Error downloading file : " +
json.getString("details")
            } else {
                def response =
getStringFromInputStream(connection.getInputStream());
                def pingURL = fetchPingUrlFromResponse(response, "Job
Status");

                getJobStatusDownload(pingURL, "GET");
            }
        } else {
            throw new Exception("Http status code: " + status);
        }
    } finally {
        if (connection != null)
            connection.disconnect();
        if (outputStream != null)
            outputStream.close();
        if (inputStream != null)
            inputStream.close();
    }
}
```



```
private void downloadContent(def downloadURL) throws Exception {
    HttpURLConnection connection = null;
    InputStream inputStream = null;
    FileOutputStream outputStream = null;

    try {
        URL url = new URL(downloadURL);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        connection.setInstanceFollowRedirects(false);
        connection.setDoOutput(true);
        connection.setUseCaches(false);
        connection.setDoInput(true);
        connection.setRequestProperty("Authorization",
            "Basic " + new
sun.misc.BASE64Encoder().encode((userName + ":" +
password).getBytes()));
        connection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
        int status = connection.getResponseCode();
        if (status == 200) {
            if (connection.getContentType() != null &&
connection.getContentType().equals("application/json")) {
                JSONObject json = new
JSONObject(getStringFromInputStream(connection.getInputStream()));
                System.out.println("Error downloading file : " +
json.getString("details"));
            } else {
                inputStream = connection.getInputStream();

                def downloadedFileName = fileName;
                if(fileName.lastIndexOf("/") != -1) {
                    downloadedFileName =
fileName.substring(fileName.lastIndexOf("/") + 1);
                }
                String ext = ".zip";
                if(connection.getHeaderField("fileExtension") !=
=null){
                    ext =
"."+connection.getHeaderField("fileExtension");
                }
                if(fileName.lastIndexOf(".") != -1 &&
fileName.lastIndexOf(".") != 0)
                    ext =
fileName.substring(fileName.lastIndexOf(".") + 1);
                outputStream = new FileOutputStream(new
File(downloadedFileName+ext));
                int bytesRead = -1;
                byte[] buffer = new byte[5 * 1024 * 1024];
                while ((bytesRead = inputStream.read(buffer)) !=
-1)
                    outputStream.write(buffer, 0, bytesRead);
            }
        }
    }
}
```

```
        System.out.println("File download completed.");
    }
} else {
    throw new Exception("Http status code: " + status);
}
} finally {
    if (connection != null)
        connection.disconnect();
    if (outputStream != null)
        outputStream.close();
    if (inputStream != null)
        inputStream.close();
}
}

private void getJobStatusDownload(def pingUrlString, def methodType)
throws Exception {
    boolean completed = false;
    while (!completed) {
        def pingResponse = executeRequest(pingUrlString, methodType,
null, "application/x-www-form-urlencoded");
        JSONObject json = new JSONObject(pingResponse);
        int status = json.getInt("status");
        if (status == -1) {
            try {
                System.out.println("Please wait...");
                Thread.sleep(20000);
            } catch (InterruptedException e) {
                completed = true;
                throw e;
            }
        } else {
            if (status > 0) {
                println "Error occurred: " + json.getString("details")
            } else {
                def downloadURL = fetchPingUrlFromResponse(pingResponse,
"Download link");
                downloadContent(downloadURL);
            }
            completed = true;
        }
    }
}
}
```

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

View and Delete Files

This table shows the REST APIs to view and delete files. Use these REST APIs to list and delete files. These REST APIs are version 11.1.2.3.600 and v2.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-10 View and Delete Files

Task	Request	REST Resource
List Files (v11.1.2.3.600)	GET	/applicationsnapshots
List Files (v2)	GET	/interop/rest/v2/files/list
Delete Files (v11.1.2.3.600)	DELETE	/applicationsnapshots/ {applicationSnapshotName}
Delete Files (v2)	DELETE	/interop/rest/v2/files/delete
Delete Files (v3)	POST	/interop/rest/v3/files/delete

List Files (v11.1.2.3.600)

This API (v11.1.2.3.600) lists the files in the Planning repository and returns information about the available file and application snapshots.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

This API provides details such as name, type, size and last modified time. Size and last modified are not available for LCM snapshots. See [About the EPM Automate Utility](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

GET /interop/rest/{api_version}/applicationsnapshots

Supported Media Types: application/json

Example of Request Body

The following shows an example of the request body in JSON format.

```
{
  "status":0,
  "items":[{
    "name":"sample.csv",
    "type":"EXTERNAL",
    "size":"18",
    "lastmodifiedtime":"1422534438000"
  },{
    "name":"snapshot1",
    "type":"LCM",
    "size":null,
    "lastmodifiedtime":null
  }],
  "details":null,
  "links":[{
    "data":null,
    "action":"GET",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots",
    "rel":"self"
  }]
}
```

Response

Table 8-11 Parameters

Name	Description
Details	Will be published in case of error with the error string
Status	See Migration Status Codes
Items	
Name	Name of the application snapshot
Type	Can be LCM or EXTERNAL Type signifies if this snapshot is for LCM or EXTERNAL. LCM indicates that the file is an LCM snapshot. EXTERNAL indicates that files are not LCM, such as Planning files.
Size	Size of the application snapshot in bytes. Available only for type EXTERNAL
Lastmodifiedtime	Time in Long value as per the last modified time of the file. Available only for type EXTERNAL
Links	Detailed information about the link
Href	Link to API call/ status API
Action	The HTTP call type
Rel	Will be self
Data	Parameters as key value pairs passed in the request

List Files Sample Code

Example 8-11 Java Sample – listFiles.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN - List all the files in PBCS
//
public void listFiles() throws Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots", serverUrl, apiVersion);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        if (json.get("items").equals(JSONObject.NULL))
            System.out.println("No files found");
        else {
            System.out.println("List of files :");
            JSONArray itemsArray =
json.getJSONArray("items");
            JSONObject jsonObj = null;
            for (int i=0; i < itemsArray.length(); i++){
                jsonObj = (JSONObject)itemsArray.get(i);
                System.out.println(jsonObj.getString("name"));
            }
        }
    }
}
//
// END - List all the files in PBCS
//
```

Example 8-12 cURL Sample– ListFiles.sh

```
funcListFiles() {
    url=$SERVER_URL/interop/rest/$API_VERSION/applicationsnapshots
    funcExecuteRequest "GET" $url

    list=`cat response.txt | jq 'select(.items != null)
| .items[].name'`
    if [[ ! -z $list ]]; then
        echo $list
    else
        echo "No files found"
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

Example 8-13 Groovy Sample– ListFiles.groovy

```
def listFiles() {
    def url;
    try {
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
applicationsnapshots")
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "GET", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        def items = object.items
        if (items == null) {
            println "No files found"
        }
        else {
            println "List of files : "
            items.each{
                println it.name
            }
        }
    } else {
        println "Error occurred while listing files"
        if (object.details != null)
            println "Error details: " + object.details
    }
}
```

Prerequisites: json.jar

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

List Files (v2)

This REST API (v2) lists the files in the repository and returns information about the available file and application snapshots.

This API provides details such as name, type, size and last modified time. Size and modified time are not available for LCM snapshots. See [About the EPM Automate Utility](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

GET /interop/rest/v2/files/list

Supported Media Types: application/json

Example URL

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
files/list
```

Response

Parameters:

Table 8-12 Parameters

Name	Description
Details	Will be published in case of error with the error string
Status	See Migration Status Codes
Items	
Name	Name of the file or application snapshot
Type	Can be LCM or EXTERNAL Type signifies if this is an LCM snapshot or not. LCM indicates that the file is an LCM snapshot. EXTERNAL indicates that the file is not an LCM snapshot, such as a Planning file.
Size	Size of the file in bytes. Available only for type EXTERNAL
Lastmodifiedtime	Last modified time of the file in milliseconds since 1970-01-01. Available only for type EXTERNAL
Links	Detailed information about the link
Href	Link to API call/ status API
Action	The HTTP call type
Rel	Will be self
Data	null

Example of Request Body

The following shows an example of the response body in JSON format.

```
{
  "status":0,
  "items":[{
    "name":"sample.csv",
    "type":"EXTERNAL",
    "size":"18",
    "lastmodifiedtime":"1422534438000"
  },{
    "name":"Artifact Snapshot",
```

```

        "type": "LCM",
        "size": null,
        "lastmodifiedtime": null
    }],
    "details": null,
    "links": [{
        "data": null,
        "action": "GET",
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/files/
list",
        "rel": "self"
    }]
}

```

List Files Sample Code

Sample cURL command

```

curl -X GET -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt -
H 'Content-Type: application/json' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/files/
list'

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Delete Files (v11.1.2.3.600)

Use the Delete Files (v11.1.2.3.600) REST API to delete a file from the Planning repository.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

Specify the filename with path separators in percent-encoding format, for example, using %5C as the encoded value for \ (file separator). For a file named `inbox/file1.csv`, pass it as `inbox%5Cfile1.csv`. If you are calling the cURL command to trigger the REST API, you can use a backslash \ for the path separator without URL encoding. See [About the EPM Automate Utility](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

DELETE /interop/rest/{api_version}/applicationsnapshots/
{applicationSnapshotName}

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-13 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
applicationSnaps hotName	Application snapshot name that needs to be deleted	Path	Yes	None

Response

Supported Media Types: application/json

Table 8-14 Parameters

Parameters	Description
Details	Published if there is an error with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to the API call
Action	The HTTP call type
Rel	Possible value: self
Data	Parameters as key value pair passed in the request

Example of Response Body

```
{
  "status":0,
  "links":[{"
    "data":null,
    "action":"DELETE",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2"
  }],
  "details":null
}
```

Delete Files Sample Code**Example 8-14 Java Sample – deleteFile.java**

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN - Delete a file in PBCS
//
public void deleteFile(String fileName) throws Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots/%s", serverUrl, apiVersion, fileName);
    String response = executeRequest(urlString, "DELETE", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0)
        System.out.println("File deleted successfully");
    else
        System.out.println("Error deleting file : " +
json.getString("details"));
}
//
// END - Delete a file in PBCS
//
```

Example 8-15 cURL Sample – DeleteFile.shPrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [Common Helper Functions for cURL](#)

```
funcDeleteFile() {
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName
    funcExecuteRequest "DELETE" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Deleted successfully"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 8-16 Groovy Sample – DeleteFile.groovy

Prerequisites: json.jar

Common Functions: [CSS Common Helper Functions for Groovy](#)

```
def deleteFile(filename) {
    def url;
    try {
        String encodedFileName = URLEncoder.encode(filename, "UTF-8");
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
applicationsnapshots/" + encodedFileName)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 )
        println "File deleted successfully"
    else {
        println "Error occurred while deleting file"
        if (object.details != null)
            println "Error details: " + object.details
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Delete Files (v2)

The Delete Files (v2) REST API deletes a file from the repository. This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

Filenames that use a backslash \ as path separators must be handled using escape characters, for example, using \\ as the value for \ (the file separator). For a file named `inbox\file1.csv`, pass it as `inbox\\file1.csv`.

For more information on deleting files, see EPM Automate Commands in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Note: To delete files using EPM Groovy rules, use the [v11.2.3.600 version of the same API](#) instead of the v2 version.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is v2.

Required Roles

Service Administrator

REST Resource

DELETE /interop/rest/v2/files/delete

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-15 Parameters

Name	Description	Type	Required	Default
fileName	Application snapshot name that needs to be deleted	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/files/delete
```

```
{
  "fileName": "inbox/file1.csv",
}
```

Response

Supported Media Types: application/json

Table 8-16 Parameters

Parameters	Description
Details	Published if there is an error with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to the API call
Action	The HTTP call type
Rel	Possible value: self
Data	Parameters as key value pair passed in the request

Example of Response Body

```
{
  "status":0,
  "links":[{"data":null,
    "action":"DELETE",
    "rel":"self",
```

```
        "href":"https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/  
files/delete"  
    }},  
    "details":null  
}
```

Sample cURL command

```
curl -X DELETE -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D  
respHeader.txt -H 'Content-Type: application/json' -d  
'{"fileName":"FILE_TO_BE_DELETED"}' 'https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/  
files/delete'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Delete Files (v3)

The Delete Files (v3) REST API deletes a file from the repository. This topic describes the simplified v3 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v3 API easier to use. This API is backwards compatible.

Filenames that use a backslash \ as path separators must be handled using escape characters; for example, using \\ as the value for \ (the file separator). For a file named `inbox\file1.csv`, pass it as `inbox\\file1.csv`.

For more information on deleting files, see *EPM Automate Commands in Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is v3.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v3/files/delete

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-17 Parameters

Name	Description	Type	Required	Default
fileName	File name to delete	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v3/files/delete
```

```
{
  "fileName": "inbox/file1.csv",
}
```

Response

Supported Media Types: application/json

Table 8-18 Parameters

Parameters	Description
Details	Published if there is an error with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to the API call
Action	The HTTP call type
Rel	Possible value: self
Data	Parameters as key value pair passed in the request

Example Response Body

```
{
  "status":0,
  "links":[{"data":null,
    "action": "POST",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v3/files/
delete"
  }],
  "details":null
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d
'{"fileName":"FILE_TO_BE_DELETED"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
v3/files/delete'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Manage Services

You can manage all available services using the following REST resources.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-19 Manage Services

Task	Request	REST Resource
Get Information About All Services	GET	GET /interop/rest/{api_version}/services
Run Recreate on a Service (11.1.2.3.600)	GET	/interop/rest/{api_version}/services/{servicename}/recreate
Run Recreate on a Service (v2)	POST	/interop/rest/v2/config/services/recreate
Restart the Service Instance (v1)	POST	/interop/rest/{api_version}/services/{service_type}/resetservice
Restart the Service Instance (v2)	POST	/interop/rest/v2/config/services/reset

Get Information About All Services

Returns information about all services that you can perform in a Planning environment.

This REST API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

GET /interop/rest/{api_version}/services

Response

Supported Media Types: application/json

Parameters:

Table 8-20 Parameters

Name	Description
api_version	Specific API version
details	In case of errors, details are published with the error string
status	See Migration Status Codes
details	In case of error, details are published with the error string
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self, PBCS recreate service, PBCS reset service - details are for PBCS recreate service
data	Parameters as key value pair passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details":null,
  "status":0,
  "links":[{"href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
services",
    "rel":"self",
    "data":null,
    "action":"GET"
  },{
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
services/PBCS/recreate",
    "rel":"PBCS recreate service",
    "data":null,
    "action":"POST"
  },{
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
services/PBCS/resetservice",
    "rel":"PBCS reset service",
    "data":null,
    "action":"POST"
  }]
}
```


Get Information About All Services Sample Code**Java Sample – getInfoAboutAllServices.java**

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN - Get services
//
public void getServices() throws Exception {
    String urlString = String.format("%s/interop/rest/%s/services",
serverUrl, apiVersion);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        JSONArray linksArray = json.getJSONArray("links");
        System.out.println("Services list :");
        JSONObject jsonObj = null;
        for(int i=0; i < linksArray.length(); i++){
            jsonObj = (JSONObject)linksArray.get(i);
            System.out.println("Service :" + jsonObj.getString("rel"));
            System.out.println("URL :" + jsonObj.getString("href"));
            System.out.println("Action :" + jsonObj.getString("action") +
"\n");
        }
    }
}
//
// END - Get services
//
```

cURL Sample – GetInfoAboutAllServices.shPrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [Common Helper Functions for cURL](#)

```
funcGetServices() {
    url=$SERVER_URL/interop/rest/$API_VERSION/services
    funcExecuteRequest "GET" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Services list :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            rel=`echo $output | jq '.links[$i].rel'`
            rel=`echo "$rel" | tr -d "\"" `
            if [ "$rel" != "self" ]; then
                echo "Service : " `echo $output | jq
```

```

'.links['$i'].rel'`
        echo "URL :" `echo $output | jq '.links['$i'].href'`
        echo "Action :" `echo $output | jq '.links['$i'].action'`
        echo ""
    fi
    i=`expr $i + 1`
done
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – GetInfoAboutAllServices.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def getServices() {
    def url;
    try {
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
services")
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "GET", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        def links = object.links
        println "Services list :"
        links.each{
            if(!it.rel.equals("self")) {
                println "Service : " + it.rel
                println "URL : " + it.href
                println "Action : " + it.action + "\n"
            }
        }
    } else {
        println "Error occurred while fetching services list"
        if (object.details != null)
            println "Error details: " + object.details
    }
}

```

Run Recreate on a Service (11.1.2.3.600)

The Run Recreate on a Service (v11.1.2.3.600) REST API restores an environment to a clean state by recreating the deployment.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

You re-create the deployment to complete these tasks:

- Clean up an environment before importing a full snapshot.
- Change the business process that can be deployed in an environment.
- Change the Essbase version in use in Oracle Enterprise Performance Management Cloud environments other than Narrative Reporting, Oracle Enterprise Data Management Cloud, and Account Reconciliation, which do not use Essbase.
By default, EPM Standard Cloud Service and EPM Enterprise Cloud Service environments are deployed with Hybrid-enabled Essbase, while legacy environments are deployed with Non-Hybrid Essbase. Downgrading the deployment of Hybrid-enabled Essbase in EPM Standard Cloud Service and EPM Enterprise Cloud Service environments is required, if you are importing a snapshot from an environment that has a Non-Hybrid Essbase version. Upgrading the deployment of Non-Hybrid Essbase in legacy environments is required to:
 - Support the extended dimensionality in existing legacy Financial Consolidation and Close environments
 - Enable hybrid block storage (BSO) applications in legacy Enterprise Planning and Planning Modules environments

For detailed information about Hybrid Essbase and the considerations for upgrading to Hybrid Essbase, see [About Essbase in EPM Cloud](#) in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

Caution:

- **This API deletes the existing application and, optionally, all user defined artifacts from the environment. Additionally, it re-creates the database and removes all existing data. After recreating the service, you can create a new business process or import one using REST APIs, Migration, or EPM Automate.**
- **This API deletes migration history. As a result, the Migration Status Report available in Migration will not contain historic information.**
- **Before using this API, perform a complete backup of the environment. You can create a backup snapshot by executing runDailyMaintenance.**

This REST API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/{api_version}/services/{servicename}/recreate

Request

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the client request.

Table 8-21 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
servicename	Name of the service for which recreate needs to be run, such as PBCS	Path	Yes	None

Table 8-21 (Cont.) Parameters

Name	Description	Type	Required	Default
tempServiceType	<p>Optionally, convert an environment to a different service environment. The business processes that you can deploy in an environment is governed by the type of subscription that you have. For example, if you have an EPM Standard Cloud Service subscription, you cannot create a FreeForm application after converting the environment from Account Reconciliation to Planning. If you have an EPM Enterprise Cloud Service subscription, you can create any business process in your environment after changing the service type appropriately. See "About the New EPM Cloud Services" in <i>Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>.</p> <p>The behavior of this parameter is dependent on your subscription. For details and examples, see Recreate in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i>.</p> <p>Acceptable tempServiceType values:</p> <ul style="list-style-type: none"> • ARCS to convert an environment to an Account Reconciliation environment • EDMCS to convert an environment to an Oracle Enterprise Data Management Cloud environment • EPRCS to convert an environment to a Narrative Reporting environment • PCMCS to convert an environment to a Profitability and Cost Management Cloud environment • PBCS to convert a Profitability and Cost Management environment to a Planning, Enterprise Planning, or Enterprise Profitability and Cost Management environment <p>Note: You can create a Tax Reporting or Financial Consolidation and Close application in a new Planning environment. You do not need to change the service type of the environment.</p> <p>When you run this REST API with tempServiceType, the serviceType change can be verified by making a REST call to /interop/rest.</p>	Payload	No	None
removeAll	If set to true, deletes all the snapshots and the content of the Inbox and Outbox folders	Payload	No	None

Table 8-21 (Cont.) Parameters

Name	Description	Type	Required	Default
essbaseChange	<p>Optionally, upgrade or downgrade the current Essbase version. The REST API retains the current Essbase version if you do not specify this option. Permissible values are:</p> <ul style="list-style-type: none"> • upgrade to change from Non-Hybrid Essbase to Hybrid Essbase • downgrade to change from Hybrid Essbase to Non-Hybrid Essbase <p>Caution: Before using this option, read and understand the information available in About Essbase in EPM Cloud in <i>Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>.</p>	Payload	No	None

Response

Parameters:

Table 8-22 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self and/or Job Status. If the value is set to Job Status, you can use the href to get the status of the recreate service
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details":null,
  "status":0,
  "links":[{"href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
services/PBCS/recreate",
    "rel":"self",
    "data":null,
    "action":"POST"
  },{
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
```

```

services/PBCS/recreate/777",
    "rel":"Job Status",
    "data":null,
    "action":"GET"
  ]]
}

```

Run Recreate on a Service Sample Code

Example 8-17 Java Sample – runRecreateOnAService.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```

//
// BEGIN - Recreate services
//
public void recreateService(String serviceName, String serviceType,
String removeAll, String essbaseChange) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.println("Are you sure you want to recreate the EPM
environment (yes/no): no ?[Press Enter]");
    String s = in.nextLine();
    if (!s.equals("yes")) {
        System.out.println("User cancelled the recreate command");
        System.exit(0);
    }

    JSONObject params = new JSONObject();
    //params.put("tempServiceType", serviceType);
    params.put("removeAll", removeAll);
    params.put("essbaseChange", essbaseChange);
    String parameters = "parameters="+
    URLEncoder.encode(params.toString(), "UTF-8");
    String urlString = String.format(
        "%s/interop/
rest/%s/services/%s/recreate", serverUrl, apiVersion, serviceName);
    String response = executeRequest(urlString, "POST", parameters,
"application/x-www-form-urlencoded");
    getJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
}

//
// END - Recreate services
//

```

Example 8-18 cURL Sample – RunRecreateOnAService.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
funcRecreateService() {
    echo "Are you sure you want to recreate the EPM environment (yes/
no): no ?[Press Enter]"
    read toCreate
    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$API_VERSION/services/EPM/recreate
    json=$(echo "{\"removeAll\":\"true\",\"essbaseChange\":\"upgrade\"}"
| sed -f urlencode.sed)
    param="parameters=$json"
    funcExecuteRequest "POST" $url $param "application/x-www-form-
urlencoded"
    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started recreating the environment successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Example 8-19 Groovy Sample – RunRecreateOnAService.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def recreateService(serviceName) {
    def toCreate = System.console().readLine 'Are you sure you want to
recreate the EPM environment (yes/no): no ?[Press Enter]'
    if (!toCreate.equals("yes")) {
        println "User cancelled the recreate command"
        System.exit(0)
    }
    def url;
    JSONObject params = new JSONObject();
    try {
        //params.put("tempServiceType", serviceType);
        params.put("removeAll", "true");
        params.put("essbaseChange", "upgrade");
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
services/" + serviceName + "/recreate");
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", "parameters="+param.toString(),
```



```
"application/x-www-form-urlencoded");
    if (response != null) {
        getJobStatus(response, "GET");
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Run Recreate on a Service (v2)

Use the Run Recreate on a Service (v2) REST API to restore an environment to a clean state by recreating the deployment.

This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

You re-create the deployment to complete these tasks:

- Clean up an environment before importing a full snapshot.
- Change the business process that can be deployed in an environment.
- Change the Essbase version in use in Oracle Enterprise Performance Management Cloud environments other than Narrative Reporting, Oracle Enterprise Data Management Cloud, and Account Reconciliation, which do not use Essbase.

By default, EPM Standard Cloud Service and EPM Enterprise Cloud Service environments are deployed with Hybrid-enabled Essbase, while legacy environments are deployed with Non-Hybrid Essbase. Downgrading the deployment of Hybrid-enabled Essbase in EPM Standard Cloud Service and EPM Enterprise Cloud Service environments is required, if you are importing a snapshot from an environment that has a Non-Hybrid Essbase version. Upgrading the deployment of Non-Hybrid Essbase in legacy environments is required to:

- Support the extended dimensionality in existing legacy Financial Consolidation and Close environments
- Enable hybrid block storage (BSO) applications in legacy Enterprise Planning and Planning Modules environments

For detailed information about Hybrid Essbase and the considerations for upgrading to Hybrid Essbase, see [About Essbase in EPM Cloud](#) in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

Caution:

- **This API deletes the existing application and, optionally, all user defined artifacts from the environment. Additionally, it re-creates the database and removes all existing data. After recreating the service, you can create a new**

business process or import one using REST APIs, Migration, or EPM Automate.

- This API deletes migration history. As a result, the Migration Status Report available in Migration will not contain historic information.
- Before using this API, perform a complete backup of the environment. You can create a backup snapshot by executing `runDailyMaintenance`.

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v2/config/services/recreate`

Request

Supported Media Types: `application/json`

Parameters:

The following table summarizes the client request.

Table 8-23 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version	Path	Yes	None

Table 8-23 (Cont.) Parameters

Name	Description	Type	Required	Default
tempServiceType	<p>Optionally, convert an environment to a different service environment. The business processes that you can deploy in an environment is governed by the type of subscription that you have. For example, if you have an EPM Standard Cloud Service subscription, you cannot create a FreeForm application after converting the environment from Account Reconciliation to Planning. If you have an EPM Enterprise Cloud Service subscription, you can create any business process in your environment after changing the service type appropriately. See "About the New EPM Cloud Services" in <i>Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>.</p> <p>The behavior of this parameter is dependent on your subscription. For details and examples, see Recreate in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i>.</p> <p>Acceptable tempServiceType values:</p> <ul style="list-style-type: none"> • ARCS to convert an environment to an Account Reconciliation environment • EDMCS to convert an environment to an Oracle Enterprise Data Management Cloud environment • EPRCS to convert an environment to a Narrative Reporting environment • PCMCS to convert an environment to a Profitability and Cost Management Cloud environment • PBCS to convert a Profitability and Cost Management environment to a Planning, Enterprise Planning, or Enterprise Profitability and Cost Management environment <p>Note: You can create a Tax Reporting or Financial Consolidation and Close application in a new Planning environment. You do not need to change the service type of the environment.</p> <p>When you run this REST API with tempServiceType, the serviceType change can be verified by making a REST call to /interop/rest.</p>	Payload	No	None
removeAll	If set to true, deletes all the snapshots and the content of the Inbox and Outbox folders	Payload	No	None

Table 8-23 (Cont.) Parameters

Name	Description	Type	Required	Default
essbaseChange	<p>Optionally, upgrade or downgrade the current Essbase version. The REST API retains the current Essbase version if you do not specify this option. Permissible values are:</p> <ul style="list-style-type: none"> • upgrade to change from Non-Hybrid Essbase to Hybrid Essbase • downgrade to change from Hybrid Essbase to Non-Hybrid Essbase <p>Caution: Before using this option, read and understand the information available in About Essbase in EPM Cloud in <i>Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>.</p>	Payload	No	None

Request

The payload is JSON.

Example of request payload:

```
{
  "parameters": {
    "removeAll": "true",
    "tempServiceType": "ARCS",
    "essbaseChange": "default"
  }
}
```

Response

Parameters:

Table 8-24 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <i>self</i> and/or <i>Job Status</i> . If the value is set to <i>Job Status</i> , you can use the href to get the status of the recreate service
data	Parameters as key value pairs passed in the request

Example of Response Body in JSON format:

```
{
  "details": null,
  "status": 0,
  "links": [{
    "href": "https://<URL>/interop/rest/v2/config/services/
recreate",
    "rel": "self",
    "data": null,
    "action": "POST"
  },
  {
    "href": "https://<URL>/interop/rest/v2/config/status/service/
recreate/1",
    "rel": "Job Status",
    "data": null,
    "action": "GET"
  }
  ]
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d '{"parameters":
{"removeAll": "false", "essbaseChange": "upgrade",
"tempServiceType": "PCMCS"}"' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
config/services/recreate'
```

Sample cURL code

```
#!/bin/sh

USERNAME="<USERNAME>"
PASSWORD="<PASSWORD>"
SERVER_URL="<SERVICE_URL>"

APP_NAME="Vision"
API_VERSION="v2"

funcRemoveTempFiles() {
  for var in "$@"
  do
    if [ -f $var ]; then
      rm $var
    fi
  done
}

funcPrintErrorDetails() {
  contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -d
[:space:]`
  if [ ! -z $contentType ] && [[ $contentType = *application/
```

```
json"* ]]; then
    output=`cat $1`
    error=`echo $output | jq '.details'`
    echo "Error details: " $error
fi
}

funcExecuteRequest() {
    if [ ! -z "$4" ]; then
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $4" -d "$3" $2`
    else
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $3" $2`
    fi

    if [ $statusCode != 200 ]; then
        echo "Error executing request"
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails "response.txt"
            funcRemoveTempFiles "respHeader.txt" "response.txt"
        fi
        exit 0
    fi
}

funcGetStatus() {
    output=`cat response.txt`
    count=`echo $output | jq '.links | length'`
    i=0
    pingUrl=""

    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links[$i].rel'`
        rel=`echo "$rel" | tr -d "\""`
        if [ "$rel" == "Job Status" ]; then
            pingUrl=`echo $output | jq '.links[$i].href'`
            pingUrl=`echo "$pingUrl" | tr -d "\""`
        fi
        i=`expr $i + 1`
    done

    echo $pingUrl
    completed="false"

    while [ $completed != "true" ]; do
        statusCode2=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "pingResponse.txt" -H "Content-Type: application/
json" "$pingUrl`
        if [ $statusCode2 == 200 ]; then
            status2=`jq '.status' pingResponse.txt`
            if [ $status2 != -1 ]; then
                completed="true"
                echo "Job completed"
            fi
        fi
    done
}
```

```
        else
            echo "Please wait..."
            sleep 20
        fi
    else
        echo "Please wait..."
        sleep 20
    fi
    funcRemoveTempFiles "pingResponse.txt"
done
}

funcHardReset() {
    echo "Are you sure you want to restart the service instance (yes/
no): no? [Press Enter] "
    read toReset
    if [ $toReset != "yes" ]; then
        echo "User cancelled the reset command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$API_VERSION/config/services/reset
    comment=$(echo $1)
    param="{\"comment\": \"${comment}\", \"parameters\":
{\"autotune\": \"false\"}}"
    funcExecuteRequest "POST" "$url" "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`

    if [ "${status}" == -1 ]; then
        echo "Started hard reset succesfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi

    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcRecreateService() {

    removeAll=$1
    essbaseChange=$2
    tempServiceType=$3

    echo "Are you sure you want to recreate the EPM environment (yes/
no): no? [Press Enter] "
    read toCreate

    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi
}
```

```

url=$SERVER_URL/interop/rest/$API_VERSION/config/services/recreate
param="{\"parameters\":{\"removeAll\": \"${removeAll}
\", \"essbaseChange\": \"${essbaseChange}\", \"tempServiceType\": \"${
tempServiceType}\"}"

funcExecuteRequest "POST" "$url" "$param" "application/json"
output=`cat response.txt`
status=`echo $output | jq '.status'`

if [ $status == -1 ]; then
    echo "Started recreating the environment successfully"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi

funcRemoveTempFiles "respHeader.txt" "response.txt"
}

if [[ "$#" != "1" ]]; then
    echo "Mandatory argument missing"
    echo "Usage: EPMRestSamples <option>"
    echo " where <option> is -recreate or -reset"
    exit 1
fi

if [ "${1}" == "-reset" ]; then
    funcHardReset "POC Exit Criteria Check - cURL"
elif [ "${1}" == "-recreate" ]; then
    funcRecreateService "false" "default" ""
else
    echo "Incorrect usage"
    echo "Usage: EPMRestSamples <option>"
    echo " where <option> is -recreate or -reset"
    exit 1
fi

```

Sample Java Code

```

package com.oracle.test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Scanner;
import java.util.Base64;

import org.json.JSONArray;

```



```
import org.json.JSONObject;

/*
 * EPM Rest Samples.
 * The userName variable uses the format <domain>.<username>.
 */
public class EPMRestSamples {

    private String userName;           // EPMCloud user name
    private String password;          // EPMCloud user password
    private String serverUrl;         // EPMCloud server URL
    private String apiVersion="v2";   // Version of the EPMCloud
Rest API

    private long startTime;
    private long endTime;
    private long maxLoopTime=(60 * 60 * 1000);

    public static void main(String[] args) {
        try {

            if(null == args || args.length != 1) {
                System.err.println("Mandatory argument missing");
                System.err.println("Usage: EPMRestSamples <option>");
                System.err.println("  where <option> is -recreate or -
reset");
                System.exit(1);
            }

            // TODO: Use appropriate username, password, and URL
            EPMRestSamples samples = new EPMRestSamples(
                "<USERNAME>", "<PASSWORD>", "<SERVICE_URL>");

            String option = args[0];
            if("-reset".equalsIgnoreCase(option)) {
                samples.hardReset("POC Exit Criteria Check - Java");
            }
            else if("-recreate".equalsIgnoreCase(option)) {
                samples.recreateService("false", "default", "");
            }
            else {
                System.err.println("Incorrect usage");
                System.err.println("Usage: EPMRestSamples <option>");
                System.err.println("  where <option> is -recreate or -
reset");
                System.exit(1);
            }
        }
        catch (Throwable x) {
            System.err.println("Error: " + x.getMessage());
        }
    }

    public EPMRestSamples(String userName, String password, String
```

```
serverUrl) throws Exception {
    this.userName = userName;
    this.password = password;
    this.serverUrl = serverUrl;
}

private String getStringFromInputStream(InputStream is) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();
    String line;

    try {
        br = new BufferedReader(new InputStreamReader(is));
        while ((line = br.readLine()) != null) {
            sb.append(line);
        }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    finally {
        if (br != null) {
            try { br.close(); }
            catch (IOException e) { e.printStackTrace(); }
        }
    }
    return sb.toString();
}

private String executeRequest(String urlString, String requestMethod,
String payload, String contentType) throws Exception {
    HttpURLConnection connection = null;
    try {
        URL url = new URL(urlString);
        Base64.Encoder encoder = Base64.getEncoder();
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod(requestMethod);
        connection.setInstanceFollowRedirects(false);
        connection.setDoOutput(true);
        connection.setUseCaches(false);
        connection.setDoInput(true);
        connection.setRequestProperty("Authorization", "Basic " +
encoder.encodeToString((userName + ":" + password).getBytes()));
        connection.setRequestProperty("Content-Type", contentType);

        if (payload != null) {
            OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
            writer.write(payload);
            writer.flush();
        }

        int status = connection.getResponseCode();
        if (status == 200 || status == 201) {
            return getStringFromInputStream(connection.getInputStream());
        }
    }
}
```

```
        }

        throw new Exception("Http status code: " + status);
    }
    finally {
        if (connection != null) { connection.disconnect(); }
    }
}

private void getJobStatus(String pingUrlString, String methodType)
throws Exception {
    boolean completed = false;

    while (!completed) {

        String pingResponse = null;
        try {
            pingResponse = executeRequest(pingUrlString,
methodType, null, "application/json");
        }
        catch (Exception e) {
            if(e instanceof java.net.ConnectException || e
instanceof java.net.SocketException) {
                if(System.currentTimeMillis()<endTime) {
                    System.out.println("Processing. Please
wait...");

                    Thread.sleep(60000);
                    continue;
                }
                throw new Exception("Command timeout..");
            }
            throw e;
        }

        JSONObject json = new JSONObject(pingResponse);
        int status = json.getInt("status");

        if (status == -1) {
            try {
                System.out.println("Processing. Please wait...");
                Thread.sleep(20000);
            }
            catch (InterruptedException ie) {
                completed = true;
                throw ie;
            }
        }
        else {
            if (status > 0) {
                System.out.println("Error occurred: " +
json.getString("details"));
            }
            else {
                System.out.println("Execution completed
successfully");
            }
        }
    }
}
```

```
        }
        completed = true;
    }
}

public String fetchPingUrlFromResponse(String response, String retValue)
throws Exception {
    String pingUrlString = null;
    JSONObject jsonObj = new JSONObject(response);
    int resStatus = jsonObj.getInt("status");

    if (resStatus == -1) {
        JSONArray lArray = jsonObj.getJSONArray("links");
        for (int i = 0; i < lArray.length(); i++) {
            JSONObject arr = lArray.getJSONObject(i);
            if (arr.get("rel").equals(retValue))
                pingUrlString = (String) arr.get("href");
        }
    }

    return pingUrlString;
}

public void hardReset(String comment) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.print("Are you sure you want to restart the service
instance (yes/no): no? [Press Enter] ");
    String s = in.nextLine();

    if (!s.equals("yes")) {
        System.out.println("User cancelled the recreate command");
        System.exit(0);
    }

    JSONObject params = new JSONObject();
    params.put("comment", comment);
    JSONObject innerParams = new JSONObject();
    innerParams.put("autotune", "true");
    params.put("parameters", innerParams);

    String urlString = String.format("%s/interop/rest/%s/config/services/
reset", serverUrl, apiVersion);
    startTime=System.currentTimeMillis();
    endTime = startTime+maxLoopTime;
    String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),"GET");
}

public void recreateService(String removeAll, String essbaseChange,
String tempServiceType) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.print("Are you sure you want to recreate the EPM
environment (yes/no): no ?[Press Enter] ");
}
```

```
String s = in.nextLine();

if (!s.equals("yes")) {
    System.out.println("User cancelled the recreate command");
    System.exit(0);
}

JSONObject params = new JSONObject();
JSONObject innerParams = new JSONObject();

innerParams.put("tempServiceType", tempServiceType);
innerParams.put("essbaseChange", essbaseChange);
innerParams.put("removeAll", removeAll);
params.put("parameters", innerParams);

String urlString = String.format("%s/interop/rest/%s/config/
services/recreate", serverUrl, apiVersion);
startTime=System.currentTimeMillis();
endTime = startTime+maxLoopTime;
String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"GET");
}
}
```

Sample Groovy Code

```
package com.groovy

import org.json.JSONObject
import groovy.json.JsonSlurper

// TODO: Use appropriate username, password, and url
username="<USERNAME>"
password="<PASSWORD>"
serverUrl="<SERVICE_URL>"

endTime=0
maxLoopTime=(60 * 60 * 1000)

apiVersion = "v2"
userCredentials = username + ":" + password
basicAuth = "Basic " + userCredentials.bytes.encodeBase64().toString()

def getResponse(is) {
    BufferedReader br = new BufferedReader(new InputStreamReader(is))
    StringBuilder sb = new StringBuilder()
    String line

    while ((line = br.readLine()) != null) {
        sb.append(line+"\n")
    }
}
```

```
        br.close()
        return sb.toString()
    }

    def getJobStatus(pingUrlString, methodType) {
        def pingUrl = new URL(pingUrlString)
        def completed = false

        while (!completed) {
            try {
                pingResponse = executeRequest(pingUrl, methodType, null,
                "application/json")
            }
            catch(exp) {
                if(exp instanceof java.net.ConnectException || exp instanceof
                java.net.SocketException) {
                    if(System.currentTimeMillis() < endTime) {
                        println("Processing. Please wait...")
                        Thread.sleep(60000)
                        continue
                    }
                    throw new Exception("Command timeout..")
                }
            }

            status = getJobStatusFromResponse(pingResponse)
            if (status == "Processing") {
                try {
                    println "Processing. Please wait..."
                    Thread.sleep(5000)
                }
                catch (InterruptedException e) {
                    completed = true
                }
            }
            else {
                println "Execution completed successfully"
                completed = true
            }
        }
    }

    def getJobStatusFromResponse(response) {
        def object = new JsonSlurper().parseText(response)
        def status = object.status
        if (status == -1) { return "Processing" }
        else if (status == 0) { return "Completed" }
        else { return object.details }
    }

    def executeRequest(url, requestType, payload, contentType) throws Exception {
        HttpURLConnection connection = (HttpURLConnection) url.openConnection()
        connection.setDoOutput(true)
        connection.setInstanceFollowRedirects(false)
        connection.setRequestMethod(requestType)
    }
}
```

```
connection.setRequestProperty("Content-Type", contentType)
connection.setRequestProperty("Authorization", basicAuth)
connection.setUseCaches(false)

if (payload != null) {
    OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream())
    writer.write(payload)
    writer.flush()
}

int statusCode
try { statusCode = connection.responseCode }
catch (all) { throw all }

def response
if (statusCode == 200 || statusCode == 201) {
    if (connection.getContentType() != null && !
connection.getContentType().startsWith("application/json")) {
        println "Error occurred in server"
        System.exit(0)
    }
    InputStream is = connection.getInputStream()
    if (is != null) { response = getResponse(is) }
}
else {

    if (statusCode == 503) {
        throw new Exception("Service Unavailable")
    }

    InputStream is = connection.getErrorStream()
    if (is != null && connection.getContentType() != null &&
connection.getContentType().startsWith("application/
json")) {
        println getJobStatusFromResponse(getResponse(is))
    }
}

connection.disconnect()
return response
}

def getUrlFromResponse(scenario, response, relValue) {
def object = new JsonSlurper().parseText(response)
def pingUrlStr
if (object.status == -1) {
    println "Started - " + scenario
    def links = object.links
    links.each{
        if (it.rel.equals(relValue)) {
            pingUrlStr=it.href
        }
    }
}
}
```

```
    else {
        println "Error details: " + object.details
        System.exit(0)
    }
    return pingUrlStr
}

def hardReset(comment) {

    def scenario = "Hard reset"
    def toReset = System.console().readLine 'Are you sure you want to
restart the service instance (yes/no): no? [Press Enter] '

    if (!toReset.equals("yes")) {
        println "User cancelled the resetService command"
        System.exit(0)
    }

    def url
    JSONObject params = new JSONObject()
    JSONObject innerParams = new JSONObject()

    try {
        params.put("comment", comment)
        innerParams.put("autotune", "true")
        params.put("parameters", innerParams)
        url = new URL(serverUrl+"/interop/rest/" + apiVersion + "/config/
services/reset")
    }
    catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0)
    }

    endTime=System.currentTimeMillis() +maxLoopTime
    response = executeRequest(url, "POST", params.toString(), "application/
json")

    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"),"GET")
    }
}

def recreateService(removeall,essabaseoption,tempServiceType) {

    def scenario="Recreate"
    def toCreate = System.console().readLine 'Are you sure you want to
recreate the EPM environment (yes/no): no? [Press Enter] '
    if (!toCreate.equals("yes")) {
        println "User cancelled the recreate command"
        System.exit(0)
    }

    def url
```



```

JSONObject params = new JSONObject()
JSONObject innerParams = new JSONObject()

try {
    innerParams.put("tempServiceType", tempServiceType)
    innerParams.put("essbaseChange", essbaseoption)
    innerParams.put("removeAll", removeall)
    params.put("parameters", innerParams)
    url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
config/services/recreate")
}
catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0)
}

endTime=System.currentTimeMillis() +maxLoopTime
response = executeRequest(url, "POST", params.toString(),
"application/json")

if (response != null) {
    getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET")
}
}

if(this.args == null || this.args.length != 1) {
    println "Mandatory argument missing"
    println "Usage: EPMRestSamples <option>"
    println "  where <option> is -recreate or -reset"
    System.exit(1)
}

def option = this.args[0]

if("-reset".equalsIgnoreCase(option)) {
    hardReset("POC Exit Criteria Check - Groovy");
}
else if("-recreate".equalsIgnoreCase(option)) {
    recreateService("false", "default", "");
}
else {
    println "Incorrect usage"
    println "Usage: EPMRestSamples <option>"
    println "  where <option> is -recreate or -reset"
    System.exit(1)
}
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Restart the Service Instance (v1)

Use the Restart the Service Instance (v1) REST API to restart the service instance with a REST API.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

You can also use an optional AutoTune parameter to auto-tune the environment before restarting it to ensure that Essbase index caches for Block Storage Option (BSO) cubes are optimized for your application.

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/{api_version}/services/{service_type}/resetservice

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 8-25 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
service_type	Value for the service type. Service type can be one of the following: PBCS for Planning and Budgeting, EPBCS for Enterprise Planning and Budgeting, FCCS for Financial Consolidation and Close, TRCS for Tax Reporting, EDMCS for Enterprise Data Management, PCMCS for Profitability and Cost Management, or ARCS for Account Reconciliation.	Path	Yes	None
comment	Comment about executing the restart	Payload	Yes	None

Table 8-25 (Cont.) Parameters

Name	Description	Type	Required	Default
autotune	If set to true, runs the auto tune algorithm before the restart. This ensures that Essbase index caches for BSO cubes are optimized for your application.	Payload	No	false

Request

The payload is JSON.

Example of request body:

```
{
  "comment": "Reset requested by Administrator",
  "autotune": "true"
}
```

Response

Table 8-26 Parameters

Name	Description
Details	In case of errors, details are published with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to API call or status API
Action	The HTTP call type
Rel	Possible values: self and/or Job Status. If the value is set to Job Status, you can use the href to get the status of the reset service
Data	Parameters as key value pairs passed in the request

The following is an example of the response body in JSON format for Planning.

```
{
  "details":null,
  "status":0,
  "links":[{"href":"https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/PBCS/resetservice",
    "rel":"self",
    "data":null,
    "action":"POST"}],{
```

```

        "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
PBCS/resetservice/777",
        "rel":"Job Status",
        "data":null,
        "action":"GET"
    ]]
}

```

Restart the Service Sample Code

Example 8-20 Java Sample – ResetServices.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```

//
// BEGIN - Reset services
//
public void hardReset(String comment) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.println("Are you sure you want to restart the service
instance (yes/no): no ?[Press Enter]");
    String s = in.nextLine();
    if (!s.equals("yes")) {
        System.out.println("User cancelled the reset command");
        System.exit(0);
    }

    JSONObject params = new JSONObject();
    params.put("comment", java.net.URLEncoder.encode(comment));
    params.put("autotune", "true");
    String urlString = String.format("%s/interop/rest/%s/services/PBCS/
resetservice", serverUrl, lcmVersion);
    String response = executeRequest(urlString, "POST", params.toString(),
"application/json");
    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"), "GET");
}
//
// END - Reset services
//

```

Example 8-21 cURL Sample – ResetServices.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

funcHardReset() {
    echo "Are you sure you want to restart the service instance (yes/no):
no ?[Press Enter] "
    read toCreate
    if [ $toCreate != "yes" ]; then
        echo "User cancelled the reset command"
    fi
}

```

```

        exit 0
    fi

    url=${SERVER_URL}/interop/rest/$LCM_VERSION/services/PBCS/
    resetservice
    comment=$(echo $1 | sed -f urlencode.sed)
    param="{\"comment\": \"$comment\", \"autotune\": \"true\"}"
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started hard reset succesfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 8-22 Groovy Sample – ResetServices.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def hardReset(comment) {
    def userInput = System.console().readLine 'Are you sure you want
    to restart the service instance (yes/no): no ?[Press Enter] '
    if (userInput.equals("yes")) {
        def url;
        JSONObject params = new JSONObject();
        try {
            params.put("comment", comment);
            params.put("autotune", "true");
            url = new URL(serverUrl + "/interop/rest/" + lcmVersion +
"/services/PBCS/resetservice");
        } catch (MalformedURLException e) {
            println "Malformed URL. Please pass valid URL"
            System.exit(0);
        }
        response = executeRequest(url, "POST", params.toString(),
"application/json");

        response = executeRequest(url, "POST", payload);
        if (response != null) {
            getJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
        }
    } else {
        println "User cancelled the resetservice command"
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Restart the Service Instance (v2)

Use the Restart the Service Instance (v2) REST API to restart the service instance.

This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

You can also use an optional AutoTune parameter to auto-tune the environment before restarting it to ensure that Essbase index caches for Block Storage Option (BSO) cubes are optimized for your application.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v2/config/services/reset`

Request

Supported Media Types: `application/json`

The following table summarizes the client request.

Table 8-27 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version	Path	Yes	None
<code>comment</code>	Comment about executing the restart	Payload	Yes	None
<code>autotune</code>	If set to <code>true</code> , runs the auto tune algorithm before the restart. This ensures that Essbase index caches for BSO cubes are optimized for your application.	Payload	No	<code>false</code>

Request

The payload is JSON.

Example of request body:

```
{
  "comment": "Reset requested by Administrator",
  "parameters": {
    "autotune": "false"
  }
}
```

Response

Table 8-28 Parameters

Name	Description
Details	In case of errors, details are published with the error string
Status	See Migration Status Codes
Links	Detailed information about the link
Href	Links to API call or status API
Action	The HTTP call type
Rel	Possible values: self and/or Job Status. If the value is set to Job Status, you can use the href to get the status of the reset service
Data	Parameters as key value pairs passed in the request

Example of Response Body in JSON format:

```
{
  "details": null,
  "status": 0,
  "links": [{
    "href": "https://<URL>/interop/rest/v2/config/services/reset",
    "rel": "self",
    "data": null,
    "action": "POST"
  },
  {
    "href": "https://<URL>/interop/rest/v2/config/status/service/hardreset/1",
    "rel": "Job Status",
    "data": null,
    "action": "GET"
  }
]
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt -H 'Content-Type: application/json' -d '{"comment": "<COMMENT>", "parameters": {"autotune": "false"}}' 'https://
```

```
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/  
rest/v2/config/services/reset'
```

Sample cURL code

```
#!/bin/sh

USERNAME="<USERNAME>"
PASSWORD="<PASSWORD>"
SERVER_URL="<SERVICE_URL>"

APP_NAME="Vision"
API_VERSION="v2"

funcRemoveTempFiles() {
    for var in "$@"
    do
        if [ -f $var ]; then
            rm $var
        fi
    done
}

funcPrintErrorDetails() {
    contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -d  
[:space:]`
    if [ ! -z $contentType ] && [[ $contentType = *"application/json"* ]];
then
        output=`cat $1`
        error=`echo $output | jq '.details'`
        echo "Error details: " $error
    fi
}

funcExecuteRequest() {
    if [ ! -z "$4" ]; then
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"  
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $4" -d "$3" $2`
    else
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"  
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $3" $2`
    fi

    if [ $statusCode != 200 ]; then
        echo "Error executing request"
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails "response.txt"
            funcRemoveTempFiles "respHeader.txt" "response.txt"
        fi
        exit 0
    fi
}
```



```

funcGetStatus() {
    output=`cat response.txt`
    count=`echo $output | jq '.links | length'`
    i=0
    pingUrl=""

    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links[$i].rel'`
        rel=`echo "$rel" | tr -d "\"`
        if [ "$rel" == "Job Status" ]; then
            pingUrl=`echo $output | jq '.links[$i].href'`
            pingUrl=`echo "$pingUrl" | tr -d "\"`
        fi
        i=`expr $i + 1`
    done

    echo $pingUrl
    completed="false"

    while [ $completed != "true" ]; do
        statusCode2=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "pingResponse.txt" -H "Content-Type:
application/json" "$pingUrl`
        if [ $statusCode2 == 200 ]; then
            status2=`jq '.status' pingResponse.txt`
            if [ $status2 != -1 ]; then
                completed="true"
                echo "Job completed"
            else
                echo "Please wait..."
                sleep 20
            fi
        else
            echo "Please wait..."
            sleep 20
        fi
        funcRemoveTempFiles "pingResponse.txt"
    done
}

funcHardReset() {
    echo "Are you sure you want to restart the service instance (yes/
no): no? [Press Enter] "
    read toReset
    if [ $toReset != "yes" ]; then
        echo "User cancelled the reset command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$API_VERSION/config/services/reset
    comment=$(echo $1)
    param="{\"comment\": \"${comment}\", \"parameters\":
{\"autotune\": \"false\"}}"
    funcExecuteRequest "POST" "$url" "$param" "application/json"

```

```
output=`cat response.txt`
status=`echo $output | jq '.status'`

if [ "${status}" == -1 ]; then
    echo "Started hard reset succesfully"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi

funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcRecreateService() {

    removeAll=$1
    essbaseChange=$2
    tempServiceType=$3

    echo "Are you sure you want to recreate the EPM environment (yes/no):
no? [Press Enter] "
    read toCreate

    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$API_VERSION/config/services/recreate
    param="{\"parameters\":{\"removeAll\": \"${removeAll}
\", \"essbaseChange\": \"${essbaseChange}\", \"tempServiceType\": \"${
tempServiceType}\"}"

    funcExecuteRequest "POST" "$url" "$param" "application/json"
    output=`cat response.txt`
    status=`echo $output | jq '.status'`

    if [ $status == -1 ]; then
        echo "Started recreating the environment successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi

    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

if [[ "$#" != "1" ]]; then
    echo "Mandatory argument missing"
    echo "Usage: EPMRestSamples <option>"
    echo " where <option> is -recreate or -reset"
    exit 1
fi
```

```
if [ "${1}" == "-reset" ]; then
    funcHardReset "POC Exit Criteria Check - cURL"
elif [ "${1}" == "-recreate" ]; then
    funcRecreateService "false" "default" ""
else
    echo "Incorrect usage"
    echo "Usage: EPMRestSamples <option>"
    echo "  where <option> is -recreate or -reset"
    exit 1
fi
```

Sample Java Code

```
package com.oracle.test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Scanner;
import java.util.Base64;

import org.json.JSONArray;
import org.json.JSONObject;

/*
 * EPM Rest Samples.
 * The userName variable uses the format <domain>.<username>.
 */
public class EPMRestSamples {

    private String userName;           // EPMCloud user name
    private String password;          // EPMCloud user password
    private String serverUrl;         // EPMCloud server URL
    private String apiVersion="v2";   // Version of the EPMCloud
    Rest API

    private long startTime;
    private long endTime;
    private long maxLoopTime=(60 * 60 * 1000);

    public static void main(String[] args) {
        try {

            if(null == args || args.length != 1) {
                System.err.println("Mandatory argument missing");
                System.err.println("Usage: EPMRestSamples <option>");
                System.err.println("  where <option> is -recreate or -
reset");
            }
        }
    }
}
```

```
        System.exit(1);
    }

    // TODO: Use appropriate username, password, and URL
    EPMRestSamples samples = new EPMRestSamples(
        "<USERNAME>", "<PASSWORD>", "<SERVICE_URL>");

    String option = args[0];
    if("-reset".equalsIgnoreCase(option)) {
        samples.hardReset("POC Exit Criteria Check - Java");
    }
    else if("-recreate".equalsIgnoreCase(option)) {
        samples.recreateService("false", "default", "");
    }
    else {
        System.err.println("Incorrect usage");
        System.err.println("Usage: EPMRestSamples <option>");
        System.err.println("  where <option> is -recreate or -
reset");
        System.exit(1);
    }
}
catch (Throwable x) {
    System.err.println("Error: " + x.getMessage());
}
}

public EPMRestSamples(String userName, String password, String
serverUrl) throws Exception {
    this.userName = userName;
    this.password = password;
    this.serverUrl = serverUrl;
}

private String getStringFromInputStream(InputStream is) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();
    String line;

    try {
        br = new BufferedReader(new InputStreamReader(is));
        while ((line = br.readLine()) != null) {
            sb.append(line);
        }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    finally {
        if (br != null) {
            try { br.close(); }
            catch (IOException e) { e.printStackTrace(); }
        }
    }
    return sb.toString();
}
```

```
    }

    private String executeRequest(String urlString, String
requestMethod, String payload, String contentType) throws Exception {
        HttpURLConnection connection = null;
        try {
            URL url = new URL(urlString);
            Base64.Encoder encoder = Base64.getEncoder();
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod(requestMethod);
            connection.setInstanceFollowRedirects(false);
            connection.setDoOutput(true);
            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setRequestProperty("Authorization", "Basic " +
encoder.encodeToString((userName + ":" + password).getBytes()));
            connection.setRequestProperty("Content-Type", contentType);

            if (payload != null) {
                OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
                writer.write(payload);
                writer.flush();
            }

            int status = connection.getResponseCode();
            if (status == 200 || status == 201) {
                return
getStringFromInputStream(connection.getInputStream());
            }

            throw new Exception("Http status code: " + status);
        }
        finally {
            if (connection != null) { connection.disconnect(); }
        }
    }

    private void getJobStatus(String pingUrlString, String methodType)
throws Exception {
        boolean completed = false;

        while (!completed) {

            String pingResponse = null;
            try {
                pingResponse = executeRequest(pingUrlString,
methodType, null, "application/json");
            }
            catch (Exception e) {
                if(e instanceof java.net.ConnectException || e
instanceof java.net.SocketException) {
                    if(System.currentTimeMillis()<endTime) {
                        System.out.println("Processing. Please
wait...");
                    }
                }
            }
        }
    }
}
```

```
        Thread.sleep(60000);
        continue;
    }
    throw new Exception("Command timeout..");
}
throw e;
}

JSONObject json = new JSONObject(pingResponse);
int status = json.getInt("status");

if (status == -1) {
    try {
        System.out.println("Processing. Please wait...");
        Thread.sleep(20000);
    }
    catch (InterruptedException ie) {
        completed = true;
        throw ie;
    }
}
else {
    if (status > 0) {
        System.out.println("Error occurred: " +
json.getString("details"));
    }
    else {
        System.out.println("Execution completed successfully");
    }
    completed = true;
}
}
}

public String fetchPingUrlFromResponse(String response, String retValue)
throws Exception {
    String pingUrlString = null;
    JSONObject jsonObj = new JSONObject(response);
    int resStatus = jsonObj.getInt("status");

    if (resStatus == -1) {
        JSONArray lArray = jsonObj.getJSONArray("links");
        for (int i = 0; i < lArray.length(); i++) {
            JSONObject arr = lArray.getJSONObject(i);
            if (arr.get("rel").equals(retValue))
                pingUrlString = (String) arr.get("href");
        }
    }

    return pingUrlString;
}

public void hardReset(String comment) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.print("Are you sure you want to restart the service
```

```
instance (yes/no): no? [Press Enter] ");
    String s = in.nextLine();

    if (!s.equals("yes")) {
        System.out.println("User cancelled the recreate command");
        System.exit(0);
    }

    JSONObject params = new JSONObject();
    params.put("comment", comment);
    JSONObject innerParams = new JSONObject();
    innerParams.put("autotune", "true");
    params.put("parameters", innerParams);

    String urlString = String.format("%s/interop/rest/%s/config/
services/reset", serverUrl, apiVersion);
    startTime=System.currentTimeMillis();
    endTime = startTime+maxLoopTime;
    String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
    getJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
    }

    public void recreateService(String removeAll, String
essbaseChange, String tempServiceType) throws Exception {
        Scanner in = new Scanner(System.in);
        System.out.print("Are you sure you want to recreate the EPM
environment (yes/no): no ?[Press Enter] " );
        String s = in.nextLine();

        if (!s.equals("yes")) {
            System.out.println("User cancelled the recreate command");
            System.exit(0);
        }

        JSONObject params = new JSONObject();
        JSONObject innerParams = new JSONObject();

        innerParams.put("tempServiceType", tempServiceType);
        innerParams.put("essbaseChange", essbaseChange);
        innerParams.put("removeAll", removeAll);
        params.put("parameters", innerParams);

        String urlString = String.format("%s/interop/rest/%s/config/
services/recreate", serverUrl, apiVersion);
        startTime=System.currentTimeMillis();
        endTime = startTime+maxLoopTime;
        String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
        getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"GET");
    }
}
```

Sample Groovy Code

```
package com.groovy

import org.json.JSONObject
import groovy.json.JsonSlurper

// TODO: Use appropriate username, password, and url
username="<USERNAME>"
password="<PASSWORD>"
serverUrl="<SERVICE_URL>"

endTime=0
maxLoopTime=(60 * 60 * 1000)

apiVersion = "v2"
userCredentials = username + ":" + password
basicAuth = "Basic " + userCredentials.bytes.encodeBase64().toString()

def getResponse(is) {
    BufferedReader br = new BufferedReader(new InputStreamReader(is))
    StringBuilder sb = new StringBuilder()
    String line

    while ((line = br.readLine()) != null) {
        sb.append(line+"\n")
    }

    br.close()
    return sb.toString()
}

def getJobStatus(pingUrlString, methodType) {
    def pingUrl = new URL(pingUrlString)
    def completed = false

    while (!completed) {
        try {
            pingResponse = executeRequest(pingUrl, methodType, null,
"application/json")
        }
        catch (exp) {
            if (exp instanceof java.net.ConnectException || exp instanceof
java.net.SocketException) {
                if (System.currentTimeMillis() < endTime) {
                    println("Processing. Please wait...")
                    Thread.sleep(60000)
                    continue
                }
            }
            throw new Exception("Command timeout..")
        }
    }

    status = getJobStatusFromResponse(pingResponse)
    if (status == "Processing") {
```



```
        try {
            println "Processing. Please wait..."
            Thread.sleep(5000)
        }
        catch (InterruptedException e) {
            completed = true
        }
    }
    else {
        println "Execution completed successfully"
        completed = true
    }
}

def getJobStatusFromResponse(response) {
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == -1) { return "Processing" }
    else if (status == 0) { return "Completed" }
    else { return object.details }
}

def executeRequest(url, requestType, payload, contentType) throws
Exception {
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection()
    connection.setDoOutput(true)
    connection.setInstanceFollowRedirects(false)
    connection.setRequestMethod(requestType)
    connection.setRequestProperty("Content-Type", contentType)
    connection.setRequestProperty("Authorization", basicAuth)
    connection.setUseCaches(false)

    if (payload != null) {
        OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream())
        writer.write(payload)
        writer.flush()
    }

    int statusCode
    try { statusCode = connection.responseCode }
    catch (all) { throw all }

    def response
    if (statusCode == 200 || statusCode == 201) {
        if (connection.getContentType() != null && !
connection.getContentType().startsWith("application/json")) {
            println "Error occurred in server"
            System.exit(0)
        }
        InputStream is = connection.getInputStream()
        if (is != null) { response = getResponse(is) }
    }
}
```

```
else {

    if (statusCode == 503) {
        throw new Exception("Service Unavailable")
    }

    InputStream is = connection.getErrorStream()
    if (is != null && connection.getContentType() != null &&
        connection.getContentType().startsWith("application/json")) {
        println getJobStatusFromResponse(getResponse(is))
    }
}

connection.disconnect()
return response
}

def getUrlFromResponse(scenario, response, relValue) {
    def object = new JsonSlurper().parseText(response)
    def pingUrlStr
    if (object.status == -1) {
        println "Started - " + scenario
        def links = object.links
        links.each{
            if (it.rel.equals(relValue)) {
                pingUrlStr=it.href
            }
        }
    }
    else {
        println "Error details: " + object.details
        System.exit(0)
    }
    return pingUrlStr
}

def hardReset(comment) {

    def scenario = "Hard reset"
    def toReset = System.console().readLine 'Are you sure you want to
restart the service instance (yes/no): no? [Press Enter] '

    if (!toReset.equals("yes")) {
        println "User cancelled the resetService command"
        System.exit(0)
    }

    def url
    JSONObject params = new JSONObject()
    JSONObject innerParams = new JSONObject()

    try {
        params.put("comment", comment)
        innerParams.put("autotune", "true")
        params.put("parameters", innerParams)
    }
```

```
        url = new URL(serverUrl+"/interop/rest/" + apiVersion + "/"
config/services/reset")
    }
    catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0)
    }

    endTime=System.currentTimeMillis() +maxLoopTime
    response = executeRequest(url, "POST", params.toString(),
"application/json")

    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"),"GET")
    }
}

def recreateService(removeall,essabaseoption,tempServiceType) {

    def scenario="Recreate"
    def toCreate = System.console().readLine 'Are you sure you want to
recreate the EPM environment (yes/no): no? [Press Enter] '
    if (!toCreate.equals("yes")) {
        println "User cancelled the recreate command"
        System.exit(0)
    }

    def url
    JSONObject params = new JSONObject()
    JSONObject innerParams = new JSONObject()

    try {
        innerParams.put("tempServiceType", tempServiceType)
        innerParams.put("essbaseChange", essabaseoption)
        innerParams.put("removeAll", removeall)
        params.put("parameters", innerParams)
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
config/services/recreate")
    }
    catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0)
    }

    endTime=System.currentTimeMillis() +maxLoopTime
    response = executeRequest(url, "POST", params.toString(),
"application/json")

    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"),"GET")
    }
}
```

```

if(this.args == null || this.args.length != 1) {
    println "Mandatory argument missing"
    println "Usage: EPMRestSamples <option>"
    println "  where <option> is -recreate or -reset"
    System.exit(1)
}

def option = this.args[0]

if("-reset".equalsIgnoreCase(option)) {
    hardReset("POC Exit Criteria Check - Groovy");
}
else if("-recreate".equalsIgnoreCase(option)) {
    recreateService("false", "default", "");
}
else {
    println "Incorrect usage"
    println "Usage: EPMRestSamples <option>"
    println "  where <option> is -recreate or -reset"
    System.exit(1)
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Manage Application Snapshots

You can manage the file system artifacts or application snapshots using the following REST resources.

Note: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the `copysnapshot` REST API. See the `encrypt` command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-29 Manage Application Snapshots

Task	Request	REST Resource
Get Information About All Application Snapshots	GET	<code>/interop/rest/{api_version}/applicationsnapshots</code>
Get Information About a Specific Application Snapshot	GET	<code>/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}</code>

Table 8-29 (Cont.) Manage Application Snapshots

Task	Request	REST Resource
Copy Application Snapshot (v1)	POST	/interop/rest/v1/services/{servicename}/copysnapshot
Copy Application Snapshot (v1)	POST	/interop/rest/v1/services/{servicename}/copysnapshot
Copy Application Snapshot (v2)	POST	/interop/rest/v2/snapshots/copyfrominstance

Get Information About All Application Snapshots

This API returns information about all application snapshots that are available in an Planning instance. It provides details such as name, type, size, and last modified time. Type signifies whether it is a Migration snapshot or an external snapshot. Size and last modified time are not available for Migration type snapshots.

This API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

GET /interop/rest/{api_version}/applicationsnapshots

Response

Supported Media Types: application/json

Parameters:

Table 8-30 Parameters

Name	Description
api_version	Specific API version
details	In case of errors, details are published with the error string
status	See Migration Status Codes
items	Detailed information about the API
name	Name of the application snapshot
type	Possible values: LCM, EXTERNAL
size	Size of the application snapshot in bytes. Available only for type EXTERNAL
lastmodifiedtime	Time in Long value as per the last modified time of the file. Will be available only for type EXTERNAL
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self

Table 8-30 (Cont.) Parameters

Name	Description
data	Parameters as key value pairs passed in the request

The following shows an example of the response body in JSON format.

```
{
  "status":0,
  "items":[{
    "name":"sample.csv",
    "type":"EXTERNAL",
    "size":"18",
    "lastmodifiedtime":"1422534438000"
  },{
    "name":"snapshot1",
    "type":"LCM",
    "size":null,
    "lastmodifiedtime":null
  }],
  "details":null,
  "links":[{
    "data":null,
    "action":"GET",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots",
    "rel":"self"
  }]
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Get Information About a Specific Application Snapshot

Returns information about all the operations that can be performed on a particular application snapshot. It provides details on operations such as Migration import and export, upload, download, and delete.

This API is version 11.1.2.3.600.

Required Roles

Service Administrator

REST Resource

GET /interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}

Request

Parameters:

The following table summarizes the GET request parameters.

Table 8-31 Parameters

Name	Description	Type	Required	Default
applicationSnapshotName	Application snapshot name to retrieve the details	Path	Yes	N/A

Response

Supported Media Types: application/json

Parameters:

Table 8-32 Parameters

Name	Description
details	In the case of an error, details are published with the error string
status	See Migration Status Codes
items	Detailed information about the API
name	Name of the application snapshot
type	Possible values: LCM, EXTERNAL
canexport	Identifies whether this application snapshot can be exported using Migration. Applicable only to Migration application artifacts
canimport	Identifies whether this application snapshot can be imported using Migration. Applicable only to Migration application artifacts
canupload	Identifies whether the application snapshot can be uploaded
candownload	Identifies whether the application snapshot can be downloaded
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values: self, import, export, upload, download, or delete depending on the operation permitted on an application snapshot
data	Parameters as key value pairs passed in the request

The following is an example of the response body in JSON format.

```
{
  "status":0,
  "items":[{"name":"snapshot1",
    "type":"LCM",
    "canexport":true,
    "canimport":true,
    "canupload":true,
    "candownload":true
```

```

    }},
    "details":null,
    "links":[{"
      "data":null,
      "action":"GET",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/snapshot1",
      "rel":"self"
    },{
      "data":null,
      "action":"GET",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/snapshot1/contents",
      "rel":"download"
    },{
      "data":null,
      "action":"POST",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/snapshot1/contents?
isLast=true&chunkSize=52428800&isFirst=true",
      "rel":"upload"
    },{
      "data":null,
      "action":"POST",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/snapshot1/migrationq={type:"export}"
      "rel":"export"
    },{
      "data":null,
      "action":"POST",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/snapshot1/migrationq={type:"import}"
      "rel":"import"
    },{
      "data":null,
      "action":"DELETE",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss1",
      "rel":"delete"
    }
  ]
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Get Information about a Specific Application Snapshot Sample Code

Example 8-23 Java Sample – `getInfoAboutSpecificSnapshots.java`

Prerequisites: `json.jar`

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN - Get application snapshot details
//
public void getApplicationSnapshotDetails(String snapshotName) throws
Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots/%s", serverUrl, apiVersion, snapshotName);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);

    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        System.out.println("Application details :");
        JSONArray itemsArray = json.getJSONArray("items");
        JSONObject item = (JSONObject) itemsArray.get(0);
        System.out.println("Application snapshot name : " +
item.getString("name"));
        System.out.println("Application snapshot type : " +
item.getString("type"));
        System.out.println("Can be exported flag : " +
item.getString("canExport"));
        System.out.println("Can be imported flag : " +
item.getString("canImport"));
        System.out.println("Can be uploaded flag : " +
item.getString("canUpload"));
        System.out.println("Can be downloaded flag : " +
item.getString("canDownload"));

        JSONArray linksArray = json.getJSONArray("links");
        JSONObject jsonObj = null;
        System.out.println("Services details :");
        for(int i=0; i < linksArray.length(); i++){
            jsonObj = (JSONObject)linksArray.get(i);
            System.out.println("Service : " + jsonObj.getString("rel"));
            System.out.println("URL : " + jsonObj.getString("href"));
            System.out.println("Action : " + jsonObj.getString("action") +
"\n");
        }
    }
}
//
// END - Get application snapshot details
//
```

Example 8-24 cURL Sample – GetInfoAboutSpecificSnapshots.shPrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [Common Helper Functions for cURL](#)

```

funcGetApplicationSnapshotDetails() {
    url=${SERVER_URL}/interop/rest/$API_VERSION/applicationsnapshots/$1
    funcExecuteRequest "GET" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Application details :"
        echo "Application snapshot name : " `echo $output | jq
'.items[0].name'`
        echo "Application snapshot type : " `echo $output | jq
'.items[0].type'`
        echo "Can be exported flag : " `echo $output | jq
'.items[0].canExport'`
        echo "Can be imported flag : " `echo $output | jq
'.items[0].canImport'`
        echo "Can be uploaded flag : " `echo $output | jq
'.items[0].canUpload'`
        echo "Can be downloaded flag : " `echo $output | jq
'.items[0].canDownload'`
        count=`echo $output | jq '.links | length'`
        i=0
        echo "Services details :"
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links['$i'].rel'`
            echo "URL : " `echo $output | jq '.links['$i'].href'`
            echo "Action : " `echo $output | jq '.links['$i'].action'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 8-25 Groovy Sample – GetInfoAboutSpecificSnapshots.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def getApplicationSnapshotDetails(applicationSnapshotName) {
    def url;
    try {
        String snapshotName = URLEncoder.encode(applicationSnapshotName,
"UTF-8");
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
applicationsnapshots/" + snapshotName)
    }
}

```

```
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}
response = executeRequest(url, "GET", null);
def object = new JsonSlurper().parseText(response)
def status = object.status
if (status == 0 ) {
    println "Application details :"
    println "Application snapshot name : " + object.items[0].name
    println "Application snapshot type : " + object.items[0].type
    println "Can be exported flag : " + object.items[0].canExport
    println "Can be imported flag : " + object.items[0].canImport
    println "Can be uploaded flag : " + object.items[0].canUpload
    println "Can be downloaded flag : " +
object.items[0].canDownload
    def links = object.links
    println "Services details :"
    links.each{
        println "Service : " + it.rel
        println "URL : " + it.href
        println "Action : " + it.action + "\n"
    }
} else {
    println "Error occurred while fetching application snapshot
details"
    if (object.details != null)
        println "Error details: " + object.details
}
}
```

Use the Application Snapshot Service

The following table explains the operations that can be performed on an application snapshot.

Note: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the `copysnapshot` REST API. See the *encrypt* command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand authentication, how to access the REST resources, and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-33 Application Snapshot Service

Task	Request	REST Resource
Upload Application Snapshot	POST	/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents?q={"isLast":false,"isFirst":true,"chunkSize":14,"fileSize":55445}
Download Application Snapshot	GET	/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents
Copy Application Snapshot (v1)	POST	/interop/rest/v1/services/{servicename}/copysnapshot
Copy Application Snapshot (v2)	POST	/interop/rest/v2/snapshots/copyfrominstance
Rename Application Snapshot (v1)	PUT	/interop/rest/v1/renamesnapshot
Rename Application Snapshot (v2)	PUT	/interop/rest/v2/snapshots/rename

Upload Application Snapshot

This API uploads an application snapshot to the Planning repository. The client needs to call upload API multiple times based on the size of file to be uploaded. The client needs to break the existing stream into number of chunks depending on the logic that each chunk size is not greater than 50 * 1024 * 1024 bytes.

This API is version v1.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents?q={"isLast":false,"isFirst":true,"chunkSize":14,"fileSize":55445}

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 8-34 Parameters

Name	Description	Type	Default
api_version	Specific API version	Path	N/A
applicationSnapshotName	Name of the application snapshot to be uploaded. A file with this name is created in the Planning repository. If a file or folder with this name exists in the repository, an error is thrown indicating that a file or folder exists.	Path	N/A
isLast	If the chunk being passed is the last one then set to true	Query	N/A
chunkSize	Size of the chunk being passed in bytes	Query	N/A

Table 8-34 (Cont.) Parameters

Name	Description	Type	Default
isFirst	If the chunk being passed is the first one and there will be subsequent requests for upload then set as true	Query	N/A
fileSize	The size of the file being uploaded	Query	N/A

Response

Supported Media Types: application/octet-stream

Parameters:

Table 8-35 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Is self, which denotes the URL of this REST API.
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status":0,
  "details":null,
  "links":[{"
    "data":null,
    "action":"POST",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/ss2.zip/contents?
q={\"isLast\":false,\"isFirst\":true,\"chunkSize\":14,\"fileSize\":55445},
    \"rel\":\"self\"
  }]}
}
```

Example of uploading with PostmanTo upload a file named `snapshot.zip` of size 12606 bytes:

Select request method as POST and Basic Authorization header for all the requests

Example Request 1 (To create the file)

URL `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=PARAMETERS -> {"isFirst":true,"chunkSize":14,"fileSize":"12606","isLast":false} // utf-8 encoded value of it`

Parameters `{"isFirst":true,"chunkSize":14,"fileSize":"12606","isLast":false} // utf-8 encoded value of it`

Example:

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=%7B%22isFirst%22%3Atrue%2C%22chunkSize%22%3A14%2C%22fileSize%22%3A%223318004%22%2C%22isLast%22%3Afalse%7D`

Example Response 1

```
{
  "links": [
    {
      "rel": "self", {
        "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=%7B%22isFirst%22:true,%22chunkSize%22:14,%22fileSize%22:%223318004%22,%22isLast%22:false%7D",
        "data": null,
        "action": "POST"
      }
    ],
    "details": null,
    "status": 0
  }
}
```

Example Request 2 (To upload the content)

URL `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=`

Parameters

`{"startRange":"0","isFirst":false,"chunkSize":12606,"isLast":false,"fileSize":"12606","endRange":"12605","chunkNo":1} // encoded value of it (Ensure the value of the parameters chunkSize and fileSize is equivalent to the total size of the file and endRange is set to fileSize - 1.)`

Example:

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=%7B%22startRange%22%3A%220%22%2C%22isFirst%22%3Afalse%2C%22chunkSize%22%3A12606%2C%22isL`

To select the file: Select tab **Body**, radio button **Binary**, **ChooseFile**, **Send**

Example Response 2

```
{
  "links": [
    {
      "rel": "self", {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/snapshot.zip/contents?
q=%7B%22startRange%22:%220%22,%22isFirst%22:false,%22chunkSize%22:12606
,%22isLast%22:false,%22fileSize%22:%2212606%22,%22endRange%22:%2212605%
22,%22chunkNo%22:1%7D",
        "data": null,
        "action": "POST"
      }
    ],
    "details": null,
    "status": 0
  }
}
```

Example Request 3 (To extract the content out of the zip file)

URL `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/applicationsnapshots/snapshot.zip/contents?q=`

Parameters

```
{"isFirst":false,"chunkSize":14,"fileSize":"12606","isLast":true} // utf-8
encoded value of it
```

Example:

```
https://<https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/snapshot.zip/contents?
q=%7B%22isFirst%22%3Afalse%2C%22chunkSize%22%3A14%2C%22fileSize%22%3A%
2212606%22%2C%22isLast%22%3Atrue%7D
```

To select the file: Select tab Body, radio button Binary, ChooseFile, Send

Example Response 3

```
{
  "links": [
    {
      "rel": "self", {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/snapshot.zip/contents?
q=%7B%22isFirst%22:false,%22chunkSize%22:14,%22fileSize%22:%223318004%
2,%22isLast%22:true%7D",
        "data": null,
        "action": "POST"
      }
    ],
  ],
}
```

```

    "details": null,
    "status": 0
  }

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Download Application Snapshot

Downloads the application snapshot from EPM repository to the local location from where client is being run. After receiving the response, if the content type is `application/json` then there would be an error on server and refer to details. Else, if it's `application/octet-stream`, then the content to be downloaded is part of the response and can read from the response body.

This API is version v1.

Required Roles

Service Administrator

REST Resource

GET `/interop/rest/{api_version}/applicationsnapshots/{applicationSnapshotName}/contents`

Parameters:

The following table summarizes the GET request parameters.

Table 8-36 Parameters

Name	Description	Type	Required	Default
<code>applicationSnapshotName</code>	Application snapshot name that needs to be uploaded	Path	Yes	N/A

Response

Supported Media Types: `application/json`

Response Header

`fileExtension` - This will have the file extension that can be used to create a file locally. Can contain values such as `zip` or `csv`.

Parameters:

Table 8-37 Parameters

Attribute	Description
<code>details</code>	Published in case of errors with the error string

Table 8-37 (Cont.) Parameters

Attribute	Description
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format in case there is an error during download.

```
{
  "details": "Not a valid file.",
  "status": 8,
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/s112.csv/contents",
    "action": "GET",
    "rel": "self",
    "data": null
  }]
}
```

Download Application Snapshot Sample Code**Java Sample – downloadAppSnapshot.java**

See Java Sample, downloadFile.java

cURL Sample – DownloadAppSnapshot.sh

See cURL Sample, DownloadFile.sh

Groovy Sample – DownloadAppSnapshot.groovy

See Groovy Sample, DownloadFile.groovy

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy Application Snapshot (v1)

Use this API (v1) to copy a snapshot from one environment (source) to another environment (target).

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

This API is executed on the target environment after details are provided for the source environment from which the snapshot is to be copied.

Prerequisites: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the `copysnapshot` REST API. See the `encrypt` command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v1/services/{servicename}/copysnapshot`

Supported Media Types: `application/x-www-form-urlencoded`

Table 8-38 Tasks for Copy Application Snapshot

Task	Request	REST Resource
Trigger <code>copysnapshot</code>	POST	<code>/interop/rest/{api_version}/services/{servicename}/copysnapshot</code>
Retrieve <code>copysnapshot</code> status	GET	<code>/interop/rest/{api_version}/services/{servicename}/copysnapshot/???</code>

Parameters:

The following table summarizes the POST request parameters.

Table 8-39 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version, such as v1	Path	Yes	None
<code>serviceName</code>	Name of the service, such as PBCS	Path	Yes	None
<code>snapshotName</code>	Name of the snapshot to be copied	Form	Yes	None
<code>userName</code>	User with access to the source instance	Form	Yes	None

Table 8-39 (Cont.) Parameters

Name	Description	Type	Required	Default
fpwd	The encrypted password for the source user to be passed as a string. The encrypted password must then be passed as one of the parameters for the REST API. For information on encrypting and generating the password.epw password file with EPM Automate, see the <i>encrypt</i> command in Command Reference in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i> .	Form	Yes	None
sourceURL	The URL of the source instance Note: This API also supports the previous name of this parameter, targetURL.	Form	Yes	None

Response

Supported Media Types: application/x-www-form-urlencoded

Parameters:

Table 8-40 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status of the copy snapshot
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/PBCS/copysnapshot",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/PBCS/copysnapshot/1502357937045",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null
}
```

Java Sample – CopySnapshot.java

```
//  
// BEGIN - copysnapshotfrominstance  
//  
public void copysnapshot() throws Exception {  
  
    String snapshotName = "SNAPSHOT NAME";  
    String srcUserName = "USER NAME";  
    String targetURL = "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com";  
    String srcEPWfilePath = "C:\\logs\\pwd.epw";  
    String srcDomain = null;  
  
    String urlString = String.format("%s/interop/rest/v1/services/PBCS/  
copysnapshot", serverUrl);  
  
    String fpwd = fetchPwdFromFile(srcEPWfilePath);  
    String params = null;  
  
    if (null == srcDomain) {  
        params = "snapshotName=" + snapshotName + "&userName=" + srcUserName +  
"&fpwd=" + fpwd + "&sourceURL=" + targetURL;  
    } else {  
        params = "snapshotName=" + snapshotName + "&userName=" + srcUserName +  
"&fpwd=" + fpwd + "&sourceURL=" + targetURL + "&dom=" + srcDomain;  
    }  
  
    String response = executeRequest(urlString, "POST", params, "application/x-  
www-form-urlencoded");  
  
    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"), "GET");  
}  
  
private String fetchPwdFromFile(String filePath) {  
    BufferedReader br = null;  
  
    try {  
        br = new BufferedReader(new FileReader(filePath));  
        String line = null;  
        String pwdString = null;  
        while ((line = br.readLine()) != null) {  
            pwdString = line;  
        }  
        return pwdString;  
    } catch (Exception e) {  
    } finally {  
        if (null != br)  
            try {  
                br.close();  
            } catch (IOException e) {  
                // TODO Auto-generated catch block
```

```

        e.printStackTrace();
    }

    }

    return null;
}
//
// END - copysnapshotfrominstance
//

```

cURL Sample – copysnapshot.sh

```

funcCopySnapshot() {
    url=$SERVER_URL/interop/rest/v1/services/PBCS/copysnapshot

    snapshotName="SNAPSHOT NAME"
    srcUserName="USER NAME"
    targetURL="https://<SERVICE NAME>-TENANT
NAME>.pbcs.<dcX>.oraclecloud.com"
    srcEPWfilePath="pwd.epw"
    srcDomain=""

    fpwd=`cat $ srcEPWfilePath`

    if [ "X" == "X$ srcDomain " ]; then
    param="snapshotName=$snapshotName&userName=$srcUserName
&fpwd=$fpwd&sourceURL=$targetURL"
    else
    param="snapshotName=$snapshotName&userName=$srcUserName
&fpwd=$fpwd&targetURL=$sourceURL&dom=$srcDomain"
    fi

    funcExecuteRequest "POST" $url $param "application/x-www-form-
urlencoded"
    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
    echo "Started Copysnapshot"
    funcGetStatus "GET"
    else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – copysnapshot.groovy

```

def copy() {
    def url;
    def params;

```

```
try {

    snapshotName = "test";
    srcUserName = "epm_default_cloud_admin";
    sourceURL = "https://<SERVICE_NAME>-TENANT
NAME>.pbcs.<dcX>.oraclecloud.com";
    srcEPWfilePath = "pwd.epw";
    fpwd = fetchPwdFromFile(srcEPWfilePath);
    srcDomain = null;

    //println fpwd
    if (null == srcDomain) {
        params = "snapshotName=" + snapshotName + "&userName=" +
srcUserName + "&fpwd=" + fpwd + "&targetURL=" + targetURL;
    } else {
        params = "snapshotName=" + snapshotName + "&userName=" +
srcUserName + "&fpwd=" + fpwd + "&targetURL=" + targetURL + "&dom=" +
srcDomain;
    }
    //println params
    url = new URL(serverUrl + "/interop/rest/v1/services/PBCS/copysnapshot");
} catch (MalformedURLException e) {
    println "Incorrect URL. Please a pass valid URL"
    System.exit(0);
}
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");

    if (response != null) {
        getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"GET");
    }
}

def fetchPwdFromFile(filePath) {
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader(filePath));
        String line = null;
        String pwdString = null;
        while ((line = br.readLine()) != null) {
            pwdString = line;
        }
        return pwdString;
    } catch (Exception e) {
    } finally {
        if (null != br)
            try {
                br.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

```

    }
    return null;
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy Application Snapshot (v2)

Use this REST API (v2) to copy a snapshot from one environment (source) to another environment (target).

This API is executed on the target environment after details are provided for the source environment from which the snapshot is to be copied.

Prerequisites: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the `copyfrominstance` REST API. See the `encrypt` command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v2/snapshots/copyfrominstance`

Supported Media Types: `application/json`

Table 8-41 Tasks for Copy Application Snapshot

Task	Request	REST Resource
Trigger <code>copysnapshot</code>	POST	<code>/interop/rest/v2/snapshots/copyfrominstance</code>
Retrieve <code>copysnapshot</code> status	GET	<code>/interop/rest/v2/status/jobs/777</code>

Parameters:

The following table summarizes the POST request parameters.

Table 8-42 Parameters

Name	Description	Type	Required	Default
snapshotName	Name of the snapshot to be copied	Payload	Yes	None
userName	User with the Service Administrator predefined role in the source instance	Payload	Yes	None
fpwd	The encrypted password for the source user to be passed as a string. For information on encrypting and generating the password.epw password file with EPM Automate, see the <i>encrypt</i> command in Command Reference in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i> .	Payload	Yes	None
sourceURL	The URL of the source instance	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/snapshots/copyfrominstance
```

```
{
  "snapshotName": "<NAME>",
  "userName": "<USERNAME>",
  "fpwd": "e0VQTUFUfWtWV3czam8xdDJ1cFZJbUVhSVQ3VWc9PS51cHcyMDE1LmFkbW",
  "sourceURL": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-43 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <i>self</i> or <i>Job Status</i> . If the value is set to <i>Job Status</i> , you can use the href to get the status of the copy snapshot
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/copyfrominstance",
    "data": null,
    "action": "POST"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/jobs/1502357937045",
    "data": null,
    "action": "GET"
  }],
  "details": null
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d
'{"snapshotName":"SNAPSHOT_TO_BE_COPIED",
"sourceURL":"SOURCE_URL","userName":"USER_NAME","fpwd":"ENCRYPTED_PASSW
ORD"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/copyfrominstance'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Rename Application Snapshot (v1)

This API renames a snapshot in EPM Cloud instances to a desired name. This gives you flexibility in naming your snapshots.

This REST API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

PUT /interop/rest/{api_version}/renamesnapshot

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 8-44 Parameters

Name	Description	Type	Required	Default
{api_version}	The version of the API, such as v1.	Path	Yes	None
snapshotName	The name of the snapshot to be renamed.	Form	Yes	None
newSnapshotName	The desired name of the existing snapshot.	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 8-45 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [{
    "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
    interop/rest/v1/renamesnapshot",
      "rel": "self",
      "data": null,
      "action": "PUT"
    }
  ],
}
```

```

    "details":null,
    "status":0
  }

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Rename Application Snapshot (v2)

The Rename Application Snapshot (v2) REST API renames a snapshot in EPM Cloud instances to a desired name. This gives you flexibility in naming your snapshots.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

PUT /interop/rest/v2/snapshots/rename

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-46 Parameters

Name	Description	Type	Required	Default
snapshotName	The name of the snapshot to be renamed.	Payload	Yes	None
newSnapshotName	The desired name of the existing snapshot.	Payload	Yes	None

Example URL and Payload

```

https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/rename

```

```

{
  "snapshotName": "Artifact Snapshot",

```

```

    "newSnapshotName": "Backup_snapshot"
  }

```

Response

Supported Media Types: application/json

Parameters:

Table 8-47 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	null

Example of Response Body

```

{
  "details": null,
  "status": 0,
  "links": [
    {
      "href": " https://<SERVICE_NAME>-
        <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/
rest/v2/snapshots/rename",
      "action": "PUT",
      "rel": "self",
      "data": null
    }
  ]
}

```

Sample cURL Command

```

curl -X PUT -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H 'Content-Type: application/json' -d '{"snapshotName":"Artifact
Snapshot" , "newSnapshotName" :
"Original snapshot"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com
/interop/rest/v2/snapshots/rename'

```

List Backups - Only for OCI (Gen 2) Environments

In OCI (Gen 2) environments, you can list available backup snapshots archived by Oracle in the Oracle Object storage Cloud.

You can then restore available backup snapshots (copy them to the environment). To restore backup snapshots in OCI (Gen 2) environments, see [Restore Backup](#). After copying the backup, you can archive it or use it to restore the current environment by yourself. With the List Backups and Restore Backup APIs, you no longer have to create a service request to request a backup from an OCI environment.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

GET /interop/rest/v2/backups/list

Response

Supported Media Types: application/json

Parameters:

Table 8-48 Parameters

Attribute	Description
details	Published in case of errors with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
intermittentStatus	Stats of each step performed; can be polled regularly from the job status URL

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
backups/list",
    "rel": "self",
```

```

        "data": null,
        "action": "GET"
    }],
    "status": 0,
    "items": ["2022-02-16T05:49:15/Artifact_Snapshot.zip",
"2022-02-18T05:44:54/Artifact_Snapshot.zip"]
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Restore Backup - Only for OCI (Gen 2) Environments

In OCI (Gen 2) environments, you can restore an available backup snapshot archived by Oracle in Oracle Object Storage (that is, copy it to the environment).

To view available backup snapshots, see [List Backups](#). If a backup snapshot is available, you can copy it to the current environment using the Restore Backup API.

After copying the backup, you can archive it or use it to restore the current environment by yourself. With the List Backups and Restore Backup APIs, you no longer have to create a service request to request a backup from an OCI environment.

This REST API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /interop/rest/v2/backups/restore

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the POST request parameters.

Table 8-49 Parameters

Name	Description	Type	Required	Default
backupName	The name of the backup snapshot, as listed in the response for List Backups	Payload	Yes	None

Table 8-49 (Cont.) Parameters

Name	Description	Type	Required	Default
targetName	The name of the backup snapshot, without an extension, in the target environment	Payload	No	If you do not specify this parameter, the backup snapshot is restored to the target environment prepended with the current timestamp, for example, 2022-03-10T06:37:48_Artifact_Snapshot.zip or 2022-03-30T06:22:35_EPRCS_Backup.tar.gz.

Example of Request Body

```
{
  "backupName": "2022-02-16T21:00:02/Artifact_Snapshot_2021-12-16T21:00:02",
  "parameters": {
    "targetName": "Backup_16Dec"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-50 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes

Table 8-50 (Cont.) Parameters

Name	Description
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status of the copy snapshot
intermittentStatus	Status of each step performed; can be polled regularly from the job status URL

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/backups/
restore",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
jobs/4534730166024804",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

View or Update the IP Allowlist - Only for OCI (Gen 2) Environments

The following table explains the tasks you can complete for viewing or updating the IP allowlist.

In OCI (Gen 2) environments, you can use a REST API to update or list the IP allowlist:

- Use the POST method with the `action` parameter set to `add` to add the IP addresses and Classless Inter-Domain Routings (CIDRs) to the allowlist of your environment.
- Use the POST method with the `action` parameter set to `remove` to remove the IP addresses and CIDRs from the allowlist of your environment.
- Use the GET method to list IP addresses and CIDRs in the allowlist of your environment.

Before using the REST resources, you must understand authentication, how to access the REST resources, and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-51 IP Allowlist

Task	Request	REST Resource
View the IP allowlist	GET	/interop/rest/epmociservice/v2/ipallowlist
Update the IP allowlist	POST	/interop/rest/epmociservice/v2/ipallowlist

View the IP Allowlist - Only for OCI (Gen 2) Environments

In OCI (Gen 2) environments, you can use a REST API to list the IP allowlist.

Use the GET method to list IP addresses and CIDRs in the allowlist of your environment.

This REST API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

GET /interop/rest/epmociservice/v2/ipallowlist

Request

Supported Media Types: application/json

Response

Supported Media Types: application/json

Parameters:

Table 8-52 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status.
items	List of IP addresses and CIDRs that are set as the allowlist for your environment.

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
epmociservice/v2/ipallowlist",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": 0,
  "items": [
    "ip_address1/24",
    "ip_address2",
    "ip_address3",
    "ip_address4",
    "ip_address5"
  ]
}
```

Update the IP Allowlist - Only for OCI (Gen 2) Environments

In OCI (Gen 2) environments, you can use a REST API to update the IP allowlist.

Use the POST method with the `action` parameter set to `add` to add the IP addresses and Classless Inter-Domain Routings (CIDRs) to the allowlist of your environment.

Use the POST method with the `action` parameter set to `remove` to remove the IP addresses and CIDRs from the allowlist of your environment.

This REST API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST `/interop/rest/epmociservice/v2/ipallowlist`

Request

Supported Media Types: `application/json`

Parameters:

The following table summarizes the POST request parameters.

Table 8-53 Parameters

Name	Description	Type	Required	Default
<code>fileName</code>	The path to a filename with a list of IP addresses and CIDRs that you want to add to or remove from the allowlist. If you want the file to be uploaded to the default location, the path is not required and you only need to specify the file name. Example: <code>IPList.txt</code>	Payload	Yes	None
<code>action</code>	Use <code>add</code> to add IP addresses and CIDRs to the allowlist of your environment. Use <code>remove</code> to remove IP addresses and CIDRs from the allowlist.	Payload	Yes	None

Examples of the Request Body

Example 1: Add to the IP allow list

```
{
  "fileName" : "IPList.txt",
  "action" : "add"
}
```

Example 2: Remove from the IP allow list

```
{
  "fileName" : "IPList.txt",
  "action" : "remove"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-54 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status of the copy snapshot

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
epmociservice/v2/ipallowlist",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
epmociservice/v2/status/jobs/2126145698194859",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy to and from the Object Store

This table shows the REST APIs to copy a file or a backup snapshot from the outbox of the current cloud environment (the source) to the Oracle Object Storage Cloud (the target). It also shows the REST APIs to copy a file or a backup snapshot from the Oracle Object Storage Cloud (the source) to the cloud environment (the target). These REST APIs are version v1 and v2.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-55 Application Snapshot Service

Task	Request	REST Resource
Copy to Object Store (v1)	POST	<code>/interop/rest/v1/services/copytoobjectstore</code>
Copy to Object Store (v2)	POST	<code>/interop/rest/v2/objectstorage/copyto</code>
Copy from Object Store (v1)	POST	<code>/interop/rest/v1/services/copyfromobjectstore</code>
Copy from Object Store (v2)	POST	<code>/interop/rest/v2/objectstorage/copyfrom</code>

Copy to Object Store (v1)

Use the Copy to Object Store (v1) REST API to copy a file or a backup snapshot from the current cloud environment (the source) to the Oracle Object Storage Cloud (the target). You can copy any file or snapshot available in the EPM Cloud. For example, if you export data to a file, the exported file is stored in the Outbox. You can then use this API to copy the file directly to Oracle Object Storage, assuming you have an account.

This topic describes the v1 version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v1/services/copytoobjectstore`

Supported Media Types: `application/x-www-form-urlencoded`

Table 8-56 Tasks for Copy to Object Store

Task	Request	REST Resource
Trigger copytoobjectstore	POST	/interop/rest/v1/services/copytoobjectstore
Retrieve copytoobjectstore status	GET	/interop/rest/v1/services/jobs/777

Parameters:

The following table summarizes the request parameters.

Table 8-57 Parameters

Name	Description	Type	Required	Default
url	<p>Oracle Object Storage Cloud bucket with an optional object name appended. The URL format without object name:</p> <pre>https:// swiftobjectstorage.region_identifier.orac lecloud.com/v1/namespace/bucket_name</pre> <p>The URL format with object name:</p> <pre>https:// swiftobjectstorage.region_identifier.orac lecloud.com/v1/namespace/bucket_name/ object_name</pre> <p>Components of this URL:</p> <ul style="list-style-type: none"> • region_identifier is an Oracle Cloud Infrastructure hosting region. • namespace is the top-level container for all buckets and objects. Each Oracle Cloud Infrastructure tenant is assigned a unique system-generated and immutable Object Storage namespace name at account creation time. Your tenancy's namespace name, for example, <code>axaxnpcrow5</code>, is effective across all regions. • bucket_name is the name of a logical container where you store your data and files. Buckets are organized and maintained under compartments. A system generated bucket name, for example, <code>bucket-20210301-1359</code> reflects the current year, month, day, and time. • object_name, optionally, is name that you want to use for the file on Oracle Object Storage Cloud. If an object name is not specified, the file will be copied with its original name. <p>For more information, see these topics in Oracle Cloud Infrastructure Documentation:</p> <ul style="list-style-type: none"> • Regions and Availability Domains • Understanding Object Storage Namespaces • Managing Buckets 	Form	Yes	None

Table 8-57 (Cont.) Parameters

Name	Description	Type	Required	Default
username	The name of the user who has access to the Object Store	Form	Yes	None
password	The password for the user who has access to the Object Store	Form	Yes	None
filepath	Name of the file (with path) to be copied to the Store. If you are copying a snapshot, do not specify the ZIP extension. Examples: Artifact Snapshot, inbox/File.txt	Form	Yes	None

Sample request payload:

```
url: https://swiftobjectstorage.<region_identifier>.oraclecloud.com/v1/epmclouddev/epm_artifact_snapshot
username: <username>
password: <password>
filepath: Artifact Snapshot
```

Response

Supported Media Types: application/json

Parameters:

Table 8-58 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body.

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
```

```

services/copytoobjectstore",
    "data": null,
    "action": "POST"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
jobs/1502357937045",
    "data": null,
    "action": "GET"
  }],
  "details": null
}

```

The password parameter value is clear text.

copyToObjectStore Sample code:

```

public void copyToObjectStore() throws Exception {

    String filepath = "FILE NAME/FILE PATH";
    String username = "USER NAME";
    String url = "https://
swiftobjectstorage.<region_identifier>.oraclecloud.com/v1/<namespace>/
<bucket_name>";
    String password = "PASSWORD";

    String urlString = String.format("%s/interop/rest/v1/services/
copytoobjectstore", serverUrl);

    String params = "url=" + url + "&userName=" + username + "&password=" +
password + "&filepath=" +filepath;

    String response = executeRequest(urlString, "POST", params,
"application/x-www-form-urlencoded");

    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"), "GET");
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy from Object Store (v1)

Use the Copy from Object Store (v1) REST API to copy a file or a backup snapshot from the Oracle Object Storage Cloud (the source) to the cloud environment (the target). If you are

copying a backup snapshot, this API copies it from the Object Storage bucket and extracts its contents in Oracle Enterprise Performance Management Cloud.

This topic describes the v1 version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v1/services/copyfromobjectstore

Supported Media Types: application/json

Table 8-59 Tasks for Copy from Object Store

Task	Request	REST Resource
Trigger copyfromobjectstore	POST	/interop/rest/v1/services/copyfromobjectstore
Retrieve copyfromobjectstore status	GET	/interop/rest/v1/services/jobs/777

Parameters:

The following table summarizes the request parameters.

Table 8-60 Parameters

Name	Description	Type	Required	Default
url	<p>The URL of the Object Store, appended with the bucket name and the name of the object to be copied. The URL format:</p> <pre>https:// swiftobjectstorage.region_identifier.orac lecloud.com/v1/namespace/bucket_name/ object_name</pre> <p>Components of this URL:</p> <ul style="list-style-type: none"> • region_identifier is a Oracle Cloud Infrastructure hosting region. • namespace is the top-level container for all buckets and objects. Each Oracle Cloud Infrastructure tenant is assigned a unique system-generated and immutable Object Storage namespace name at account creation time. Your tenancy's namespace name, for example, axaxnpcrow5, is effective across all regions. • bucket_name is the name of a logical container where you store your data and files. Buckets are organized and maintained under compartments. A system generated bucket name, for example, bucket-20210301-1359 reflects the current year, month, day, and time. • object_name is the name of the snapshot or file that you want to copy from Oracle Object Storage Cloud. <p>For more information, see these topics in Oracle Cloud Infrastructure Documentation:</p> <ul style="list-style-type: none"> • Regions and Availability Domains • Understanding Object Storage Namespaces • Managing Buckets 	Form	Yes	None
username	The name of the user who has access to the Object Store	Form	Yes	None
password	The password for the user who has access to the Object Store.	Form	Yes	None
targetfile	<p>Name of the target filename (with path) of the downloaded artifact. When copying snapshots, do not specify the ZIP extension.</p> <p>Examples: Artifact Snapshot_24_Sept_2020, inbox/File_new.txt</p>	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 8-61 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/copyfromobjectstore",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/jobs/1502357937045",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy to Object Store (v2)

Use the Copy to Object Store (v2) REST API to copy a file or a backup snapshot from the current cloud environment (the source) to the Oracle Object Storage Cloud (the target). You can copy any file or snapshot available in the EPM Cloud. For example, if you export data to a file, the exported file is stored in the Outbox. You can then use this API to copy the directly to Oracle Object Storage, assuming you have an account.

This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v2/objectstorage/copyto

Supported Media Types: application/json

Table 8-62 Tasks

Task	Request	REST Resource
Trigger copytoobjectstore	POST	/interop/rest/v2/objectstorage/copyto
Retrieve copytoobjectstore status	GET	/interop/rest/v2/status/jobs/777

Parameters:

The following table summarizes the request parameters.

Table 8-63 Parameters

Name	Description	Type	Required	Default
url	<p>The URL for Oracle Cloud Object Storage, appended with the bucket name and an optional object name. This is the URL format without the object name:</p> <pre>https:// swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/bucket_name</pre> <p>This is the URL format with the object name:</p> <pre>https:// swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/bucket_name/ object_name</pre> <p>Components of this URL:</p> <ul style="list-style-type: none"> • region_identifier is a Oracle Cloud Infrastructure hosting region. • namespace is the top-level container for all buckets and objects. Each Oracle Cloud Infrastructure tenant is assigned a unique system-generated and immutable Object Storage namespace name at account creation time. Your tenancy's namespace name, for example, <code>axaxnpcrow5</code>, is effective across all regions. • bucket_name is the name of a logical container where you store your data and files. Buckets are organized and maintained under compartments. A system-generated bucket name, for example, <code>bucket-20210301-1359</code> reflects the current year, month, day, and time. • object_name, optionally, is a name that you want to use for the file on Oracle Object Storage Cloud. If an object name is not specified, the file will be copied with its original name. <p>For more information, see these topics in Oracle Cloud Infrastructure documentation:</p> <ul style="list-style-type: none"> • Regions and Availability Domains • Understanding Object Storage Namespaces • Managing Buckets 	Payload	Yes	None
userName	The name of the user who has access to the Object Store	Payload	Yes	None
password	The password for the user who has access to the Object Store	Payload	Yes	None
filePath	<p>Name of the file (with path) to be copied to the Object Store. If you are copying a snapshot, do not specify the ZIP extension.</p> <p>Examples: Artifact Snapshot, inbox/File.txt</p>	Payload	Yes	None

Example URL and Payload:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/objectstorage/copyto
```

```
{
  "url": "https://
swiftobjectstorage.<region_identifier>.oraclecloud.com/v1/epmclouddev/
epm_artifact_snapshot",
  "userName": "username",
  "password": "password",
  "filePath": "Artifact Snapshot"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-64 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
objectstorage/copyto",
    "data": null,
    "action": "POST"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
jobs/1502357937045",
```

```

        "data": null,
        "action": "GET"
    }},
    "details": null
}

```

Sample cURL command

```

curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d
'{"url":"OBJECT_STORAGE_URL","userName":"USER_NAME","password":"PASSWOR
D","filePath":"FILEPATH/FILENAME"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
objectstorage/copyto'

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy from Object Store (v2)

Use the Copy from Object Store (v2) REST API to copy a file or a backup snapshot from the Oracle Object Storage Cloud (the source) to the cloud environment (the target). If you are copying a backup snapshot, this API copies it from the Object Storage bucket and extracts its contents in Oracle Enterprise Performance Management Cloud.

This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v2/objectstorage/copyfrom

Supported Media Types: application/json

Table 8-65 Tasks

Task	Request	REST Resource
Trigger copyfromobjectstore	POST	/interop/rest/v2/objectstorage/copyfrom

Table 8-65 (Cont.) Tasks

Task	Request	REST Resource
Retrieve copyfromobjectstore status	GET	/interop/rest/v2/status/jobs/777

Parameters:

The following table summarizes the request parameters.

Table 8-66 Parameters

Name	Description	Type	Required	Default
url	<p>The URL for Oracle Cloud Object Storage, appended with the bucket name and an optional object name. This is the URL format :</p> <pre>https:// swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/bucket_name/ object_name</pre> <p>Components of this URL:</p> <ul style="list-style-type: none"> • region_identifier is a Oracle Cloud Infrastructure hosting region. • namespace is the top-level container for all buckets and objects. Each Oracle Cloud Infrastructure tenant is assigned a unique system-generated and immutable Object Storage namespace name at account creation time. Your tenancy's namespace name, for example, axaxnpcrow5, is effective across all regions. • bucket_name is the name of a logical container where you store your data and files. Buckets are organized and maintained under compartments. A system-generated bucket name, for example, bucket-20210301-1359 reflects the current year, month, day, and time. • object_name is the name of the snapshot or file that you want to copy from Oracle Object Storage Cloud. <p>For more information, see these topics in Oracle Cloud Infrastructure Documentation:</p> <ul style="list-style-type: none"> • Regions and Availability Domains • Understanding Object Storage Namespaces • Managing Buckets 	Payload	Yes	None
userName	The name of the user who has access to the Object Store	Payload	Yes	None
password	The password for the user who has access to the Object Store.	Payload	Yes	None

Table 8-66 (Cont.) Parameters

Name	Description	Type	Required	Default
targetFile	Name of the target filename (with path) of the downloaded artifact. When copying snapshots, do not specify the ZIP extension. Examples: Artifact Snapshot_24_Sept_2020, inbox/File_new.txt	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/objectstorage/copyfrom
```

```
{
  "url": "https://
swiftobjectstorage.<region_identifier>.oraclecloud.com/v1/namespace/
bucket_name/object_name",
  "userName": "epm_user",
  "password": "epm_password",
  "targetFile": "Artifact snapshot"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-67 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
```

```

    "status": -1,
    "items": null,
    "links": [
      {
        "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/v2/
objectstorage/copyfrom",
        "action": "POST",
        "rel": "self",
        "data": null
      },
      {
        "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/v2/status/
jobs/4003051833546274",
        "action": "GET",
        "rel": "Job Status",
        "data": null
      }
    ]
  }
}

```

Sample cURL Command

```

curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H 'Content-Type: application/json' -d
'{"url":"OBJECT_STORAGE_URL","userName":"USER_NAME","password":"PASSWORD","ta
rgetFile":"FILEPATH/FILENAME"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
objectstorage/copyfrom'

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Provide Feedback (v11.1.2.3.600)

This feedback service sends feedback or reports an issue to Oracle.

This API is version 11.1.2.3.600.

Required Roles

Service Administrator, Power User, User, Viewer

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /interop/rest/{api_version}/feedback

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 8-68 Parameters

Name	Description
details	Published in case of errors with the error string
status	See Migration Status Codes
items	Details about the resource
issueRef	Feedback reference to contact Oracle support
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body:

Supported Media Types: application/json

Example of Response Body:

The following shows an example of the response body in JSON format.

```
{
  "details":null,
  "status":0,
  "items":
  [{"issueRef":"UDR_default_fin_superuser_2015_09_14_11_10_18"}],
  "links":[{"
    "data":null,
    "action":"POST",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
{api_version}/feedback"
  }]
}
```

Provide Feedback Sample Code

Example 8-26 Java Sample – ProvideFeedback.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN - Provide Feedback
//
public void provideFeedback(String description) throws Exception {
    JSONObject params = new JSONObject();
    JSONObject config = new JSONObject();
    config.put("URL", serverUrl);
    params.put("configuration", config);
    params.put("description", description);

    String urlString = String.format("%s/interop/rest/%s/feedback",
serverUrl, lcmVersion);
    String response = executeRequest(urlString, "POST", params.toString(),
"application/json");
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        System.out.println("Feedback successful");
    } else {
        System.out.println("Error occurred: " + json.getString("details"));
    }
}
//
// END - Provide Feedback
//
```

Example 8-27 cURL Sample – ProvideFeedback.sh

```
funcProvideFeedback() {
    url=$SERVER_URL/interop/rest/$LCM_VERSION/feedback
    description=$(echo $1 | sed -f urlencode.sed)
    param="{\"configuration\":
{\"URL\": \"$SERVER_URL\"}, \"description\": \"$description\"}"
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Feedback successful"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

Example 8-28 Groovy Sample – ProvideFeedback.groovy

```
def provideFeedback(description) {
    def url;
    JSONObject params = new JSONObject();
    try {
        JSONObject config = new JSONObject();
        config.put("URL",serverUrl)
        params.put("configuration",config);
        params.put("description",description);
        url = new URL(serverUrl + "/interop/rest/" + lcmVersion + "/"
feedback");
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params.toString(),
"application/json");

    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        println "Feedback successful"
    } else {
        println "Error occurred while listing files"
        if (object.details != null)
            println "Error details: " + object.details
    }
}
```

Prerequisites: json.jar

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Provide Feedback (v2)

The Provide Feedback (v2) feedback service sends feedback or reports an issue to Oracle.

This API is version v2.

Required Roles

Service Administrator, Power User, User, Viewer

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /interop/rest/v2/services/feedback

Request

Supported Media Types: multipart/form-data

Name	Description	Type	Required	Default
fileName	The name of the zip file that you want Oracle support to use to resolve the current issue.	Multipart	Yes	None
file	Multiple files like EPM Automate scripts or Fiddler traces required to be submitted to Oracle can be zipped altogether and uploaded.	Multipart	Yes	None
configuration	Details of the client, OS, URL, and description of the issue.	Multipart	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/feedback
```

Sample Configuration

```
configuration:{"configuration":{"Operating_System": "Windows 10", "EPMAutomate_Version": "22.11.12", "Java_Vendor": "Oracle Corporation", "Java_Version": "1.8.0_341", "URL": "http://slcar287.usdvl.oraclecloud.com:12847"}, "description": "Issue description"}}
```

Response

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 8-69 Parameters

Name	Description
details	Published in case of errors with the error string
status	See Migration Status Codes

Table 8-69 (Cont.) Parameters

Name	Description
items	Details about the resource
issueRef	Feedback reference to contact Oracle support
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: <code>self</code>
data	Parameters as key value pairs passed in the request

Example of Response Body:

```
"details": null,
  "status": 0,
  "items": [
    {
      "issueRef":
"UDR_default_epm_default_cloud_admin_2022_10_11_01_47_10"
    }
  ],
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
services/feedback",
      "action": "POST",
      "rel": "self",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt
-H 'Content-Type: multipart/form-data' -F 'fileName=abc.zip' -F
'configuration={"configuration":{"Operating_System": "Windows 10",
"EPMAutomate_Version":"22.11.12","Java_Vendor":"Oracle Corporation",
"Java_Version":"1.8.0_341","URL":"https://<SERVICE_NAME>-<TENANT_NAME>.
<SERVICE_TYPE>.<dcX>.oraclecloud.com"},"description":"abc"}}' -F
'file=@/
C:/Users/abc/Sample.zip' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.
<dcX>.oraclecloud.com/interop/rest/v2/services/feedback'
```

LCM Export (v1)

Initiates a repeat export of a Migration artifact based on the settings that were used to export artifacts using the Migration artifact export screen. This REST API is version 11.1.2.3.600.

You can also use EPM Automate to automate the repeat export of artifacts.

The presence of status -1 in the response indicates that the reexport is in progress. You should use the job status URI to determine whether the reexport is complete.

If the Job completes with status 1, the task details will be mentioned in the items from which the source, destination, and URL to fetch the first set of errors is available. All issues for a particular task can be fetched in the manner of pagination. Acceptable values for `msgtype` are: `error/warn/info`; `limit` represents the number of issues requested per request, and `offset` marks the beginning number to fetch the issues.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-70 LCM Export

Task	Request	REST Resource
LCM Export	POST	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration? q={type:"export"}
Export Status	GET	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration/8
Details	GET	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration/8/0/details? limit=25&msgtype=error&of fset=0

REST Resource

POST /interop/rest/{api_version}/applicationsnapshots/
{applicationSnapshotName}/migration?q={type:"export"}

Parameters:

The following table summarizes the request parameters.

Table 8-71 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
applicationSnaps hotName	Application snapshot that has to be exported	Path	Yes	None
type	Type of migration being performed, can be export or import	Query	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 8-72 Parameters

Attribute	Description
details	In case of errors, details are published with the error string
status	See Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values. Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the re-export operation
data	Parameters as key value pairs passed in the request
items	Details about the resource
source	From where the navigation is being performed
destination	To where the navigation is being performed
name	Name of the task, usually "Task Information"
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body

The following is an example of the response body in JSON format.

Response 1 example when export is in progress:

```
{
  "status":-1,
  "links":[{"
    "data":null,
    "action":"POST",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migrationq={type:"export"}"
  }],{
```

```

        "data":null,
        "action":"POST",
        "rel":"Job Status",
        "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/8"
    }],
    "details":null
}

```

Response 2 example when export completes with errors:

```

{"status":1,
"items":[{"
  "source":"/Nasdaq/HSS-Shared Services",
  "name":"Task Information",
  "destination":"Shared Services",
  "links":[{"
    "data":null,
    "action":"GET",
    "rel":"Job Details",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=25&offset=0&msgtype=error"}]
  },
  {"source":"/Artifact Snapshot/HP-NASDAQ",
  "name":"Task Information",
  "destination":"",
  "links":[{"
    "data":null,
    "action":"GET",
    "rel":"Job Details",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/1/details?
limit=25&offse=0&msgtype=error"}]
  }],
  "details":null,
  "links":[{"
    "data":null,
    "action":"POST",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1"}
  ]}
}

```

Response 3 example when information on each task is requested:

```

{"status":0,
"items":[{"
  "msgType":"error",
  "artifact":"/Native Directory/Groups",

```

```

    "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0026 not found. Please ensure that a user exists in
the system.",
    "msgCategory":"14000: Error reported.",
    "msgCategory":"14000: Error reported."
  },{
    "msgType":"error",
    "artifact":"/Native Directory/Groups",
    "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0025 not found. Please ensure that a user exists in
the system.",
    "msgCategory":"14000: Error reported." }
  ],
  "details":null,
  "links":[{"
    "data":null,
    "action":"GET",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&msgType=error&offset=25"},
    {"data":null,
    "action":"GET",
    "rel":"next",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=27&msgType=error"},
    {"data":null,
    "action":"GET",
    "rel":"prev",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=23&msgType=error
  }}
}

```

Export Data Sample Code

Example 8-29 Java Sample – LcmExport.java

Prerequisites: json.jar

Common functions: See [Appendix A, Common Helper Functions for Java](#).

```

//
// BEGIN - Export an application snapshot
//
public void exportSnapshot(String applicationSnapshotName) throws
Exception {
    JSONObject params = new JSONObject();
    params.put("type", "export");
    String urlString = String.format("%s/interop/rest/%s/

```

```

applicationsnapshots/%s/migration?q=%s", serverUrl, apiVersion,
URLEncoder.encode(applicationSnapshotName, "UTF-8"), params.toString());
    String response = executeRequest(urlString, "POST", null);
    System.out.println("Export started successfully");
    getMigrationJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"POST");
}
//
// END - Export an application snapshot
//

```

Example 8-30 cURL Sample – LcmExport.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

funcExportSnapshot() {
    param=$(echo "{type:export}" | sed -f urlencode.sed)
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
    applicationsnapshots/$encodedFileName/migration?q=$param
    funcExecuteRequest "POST" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started exporting successfully"
        funcGetMigrationStatus "POST"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 8-31 Groovy Sample – LcmExport.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def exportSnapshot(applicationSnapshotName) {
    def url;
    try {
        String snapshotName = URLEncoder.encode(applicationSnapshotName,
"UTF-8");
        JSONObject params = new JSONObject();
        params.put("type", "export");
        url = new URL(serverUrl + "/interop/rest/" + apiVersion + "/"
applicationsnapshots/" + snapshotName + "/migration?q=" + params.toString());
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
}

```

```

    response = executeRequest(url, "POST", null, "application/x-www-
form-urlencoded");
    if (response != null) {
        getMigrationJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "POST");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

LCM Export (v2)

The LCM Export (v2) REST API initiates a repeat export of a Migration artifact based on the settings that were used to export artifacts using the Migration artifact export screen.

The presence of status -1 in the response indicates that the reexport is in progress. You should use the job status URI to determine whether the reexport is complete.

If the Job completes with status 1, the task details will be mentioned in the items from which the source, destination, and URL to fetch the first set of errors is available. All issues for a particular task can be fetched in the manner of pagination. Acceptable values for `msgtype` are: `error/warn/info`; `limit` represents the number of issues requested per request, and `offset` marks the beginning number to fetch the issues.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-73 LCM Export

Task	Request	REST Resource
LCM Export	POST	/interop/rest/v2/snapshots/export
Export Status	GET	/interop/rest/v2/status/migration/28
Details	GET	/interop/rest/v2/status/migration/28/1/details?limit=25&offset=0&msgtype=info

REST Resource

POST /interop/rest/v2/snapshots/export

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-74 Parameters

Name	Description	Type	Required	Default
SnapshotName	Application snapshot that has to be exported	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/snapshots/export
```

```
{
  "snapshotName": "Artifact Snapshot"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-75 Parameters

Attribute	Description
details	In case of errors, details are published with the error string
status	See Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values. Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the re-export operation
data	Parameters as key value pairs passed in the request
items	Details about the resource
source	From where the navigation is being performed
destination	To where the navigation is being performed
name	Name of the task, usually "Task Information"
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body

The following is an example of the response body in JSON format.

Response 1 example when export is in progress:

```

{
  "details": null,
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/export",
      "action": "POST",
      "rel": "self",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/28",
      "action": "POST",
      "rel": "Job Status",
      "data": null
    }
  ]
}

```

Response 2 example when export completes:

```

{
  "details": null,
  "status": 0,
  "items": [
    {
      "name": "Task Information",
      "source": "Shared Services",
      "destination": "/Artifact Snapshot1/HSS-Shared Services",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/28/0/details?limit=25&offset=0&msgtype=info",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    },
    {
      "name": "Task Information",
      "source": "Vision",
      "destination": "/Artifact Snapshot1/HP-Vision",

```

```
    "links": [
      {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/28/1/details?limit=25&offset=0&msgtype=info",
        "action": "GET",
        "rel": "Job Details",
        "data": null
      }
    ]
  },
  {
    "name": "Task Information",
    "source": "Document Repository",
    "destination": "/Artifact Snapshot1/DOCREP-Document Repository",
    "links": [
      {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/28/2/details?limit=25&offset=0&msgtype=info",
        "action": "GET",
        "rel": "Job Details",
        "data": null
      }
    ]
  },
  {
    "name": "Task Information",
    "source": "Calculation Manager",
    "destination": "/Artifact Snapshot1/CALC-Calculation Manager",
    "links": [
      {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/28/3/details?limit=25&offset=0&msgtype=info",
        "action": "GET",
        "rel": "Job Details",
        "data": null
      }
    ]
  },
  {
    "name": "Task Information",
    "source": "FDM Enterprise Edition",
    "destination": "/Artifact Snapshot1/FDMEE-FDM Enterprise
Edition",
    "links": [
      {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/28/4/details?limit=25&offset=0&msgtype=info",
        "action": "GET",
        "rel": "Job Details",
        "data": null
      }
    ]
  }
]
```



```

    ]
  }
],
"links": [
  {
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/28",
    "action": "GET",
    "rel": "self",
    "data": null
  }
]
}

```

Sample cURL command

```

curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H
'Content-Type: application/json' -d '{"snapshotName":"Artifact
Snapshot"}'
'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/export'

```

LCM Import (v1)

Initiates import of a Migration snapshot so that the contents of the application snapshot are imported into the application. You can complete these tasks for imported users: set a specific password for all users in the snapshot, set a unique temporary password for each user in the snapshot, and force password reset at first login.

The presence of status -1 in the response indicates that the import is in progress. You should use the job status URI to determine whether the import is complete.

If the Job completes with status 1, the task details will be mentioned in the items from which the source, destination, and URL to fetch the first set of errors is available. All issues for a particular task can be fetched in the manner of pagination. Acceptable values for msgtype are: error/warn/info; limit represents the number of issues requested per request, and offset marks the beginning number to fetch the issues.

This API is version 11.1.2.3.600.

Required Roles

Service Administrator

Identity Domain Administrator role is required to import user and predefined roles

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-76 LCM Import

Task	Request	REST Resource
LCM Import	POST	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration? q={type:"import"}
Import Status; the 9 in the resource is used as an example here	GET	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration/9
Details	GET	/interop/rest/ {api_version}/ applicationsnapshots/ {applicationSnapshotName} /migration/9/0/details? limit=25&msgtype=error&of fset=0

REST Resource

POST /interop/rest/{api_version}/applicationsnapshots/
{applicationSnapshotName}/migration?q={type:"import"}

Parameters:

The following table summarizes the request parameters.

Table 8-77 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
applicationSnapshotName	Application snapshot that is to be imported	Path	Yes	None
type	Type of migration being performed, import	Query	Yes	None
importUsers	Whether to import users; true imports users and their predefined role assignments. The import fails if the user does not have the Identity Domain Administrator role.	Query	No	false
userPassword	The default password for the imported users.	Query	No	A unique temporary password is assigned to each user
resetPassword	Whether to force reset password for the imported users on first login, true or false	Query	No	true

Response

Supported Media Types: application/json

Table 8-78 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request
items	Details about the resource
source	From where the navigation is being performed
destination	To where the navigation is being performed
name	Name of the task, usually "Task Information"
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body

The following shows an example of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "details":null,
  "status":-1,
  "links":[{"
    "data":null,
    "action":"POST",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migrationq={type:"import"}"
  },{
    "data":null,
    "action":"POST",
    "rel":"Job Status",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/2"
  }]
}
```

Response 2 example when import completes with errors:

```

{"status":1,
"items":[{"source":"/Nasdaq/HSS-Shared Services",
"name":"Task Information",
"destination":"Shared Services",
"links":[{"data":null,
"action":"GET",
"rel":"Job Details",
"href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=25&offset=0&msgtype=error"}]
},
{"source":"/Artifact Snapshot/HP-SS2",
"name":"Task Information",
"destination":"","
"links":[{"data":null,
"action":"GET",
"rel":"Job Details",
"href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/1/details?
limit=25&offset=0&msgtype=error"}]
}],
"details":null,
"links":[{"data":null,
"action":"POST",
"rel":"self",
"href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1"}
]}
}

```

Response 3 example when each type of task information is requested:

```

{"status":0,
"items":[{"msgType":"error",
"artifact":"/Native Directory/Groups",
"msgText":"EPMIE-00069: Failed to find user during group children import.
User user0026 not found. Please ensure that a user exists in the system.",
"msgCategory":"14000: Error reported.",
"msgCategory":"14000: Error reported."
},{
"msgType":"error",
"artifact":"/Native Directory/Groups",
"msgText":"EPMIE-00069: Failed to find user during group children import.
User user0025 not found. Please ensure that a user exists in the system.",

```

```

    "msgCategory":"14000: Error reported." }
  ],
  "details":null,
  "links":[{"
    "data":null,
    "action":"GET",
    "rel":"self",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&msgtype=error&offset=25"},
    {"data":null,
    "action":"GET",
    "rel":"next",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=27&msgType=error"},
    {"data":null,
    "action":"GET",
    "rel":"prev",
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=23&msgType=error
  }}
}

```

LCM Import Sample Code

Example 8-32 Java Sample – lcmlImport.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```

{"status":0,
 "items":[{"
   "msgType":"error",
   "artifact":"/Native Directory/Groups",
   "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0026 not found. Please ensure that a user exists in
the system.",
   "msgCategory":"14000: Error reported.",
   "msgCategory":"14000: Error reported."
 },{
   "msgType":"error",
   "artifact":"/Native Directory/Groups",
   "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0025 not found. Please ensure that a user exists in
the system.",
   "msgCategory":"14000: Error reported." }
 ],

```

```

"details":null,
"links":[{"
  "data":null,
  "action":"GET",
  "rel":"self",
  "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&msgtype=error&offset=25"},
  {"data":null,
  "action":"GET",
  "rel":"next",
  "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=27&msgType=error"},
  {"data":null,
  "action":"GET",
  "rel":"prev",
  "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=23&msgType=error
  }}
}

```

Example 8-33 cURL Sample – LcmImport.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

{"status":0,
"items":[{"
  "msgType":"error",
  "artifact":"/Native Directory/Groups",
  "msgText":"EPMIE-00069: Failed to find user during group children import.
User user0026 not found. Please ensure that a user exists in the system.",
  "msgCategory":"14000: Error reported.",
  "msgCategory":"14000: Error reported."
},{
  "msgType":"error",
  "artifact":"/Native Directory/Groups",
  "msgText":"EPMIE-00069: Failed to find user during group children import.
User user0025 not found. Please ensure that a user exists in the system.",
  "msgCategory":"14000: Error reported." }
],
"details":null,
"links":[{"
  "data":null,
  "action":"GET",
  "rel":"self",
  "href":"https://<SERVICE_NAME>-

```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&msgtype=error&offset=25"},
  {"data":null,
   "action":"GET",
   "rel":"next",
   "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=27&msgType=error"},
  {"data":null,
   "action":"GET",
   "rel":"prev",
   "href":https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=23&msgType=error
  }}
}

```

Example 8-34 Groovy Sample – LcmImport.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

{"status":0,
 "items":[{"msgType":"error",
           "artifact":"/Native Directory/Groups",
           "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0026 not found. Please ensure that a user exists in
the system.",
           "msgCategory":"14000: Error reported.",
           "msgCategory":"14000: Error reported."
        },{
           "msgType":"error",
           "artifact":"/Native Directory/Groups",
           "msgText":"EPMIE-00069: Failed to find user during group children
import. User user0025 not found. Please ensure that a user exists in
the system.",
           "msgCategory":"14000: Error reported."
        }
    ],
 "details":null,
 "links":[{"data":null,
           "action":"GET",
           "rel":"self",
           "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/ss2/migration/1/0/details?
limit=2&msgtype=error&offset=25"},
          {"data":null,

```

```

    "action": "GET",
    "rel": "next",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=27&msgType=error"},
    {"data": null,
     "action": "GET",
     "rel": "prev",
     "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/rest/11.1.2.3.600/
applicationsnapshots/ss2/migration/1/0/details?
limit=2&offset=23&msgType=error
    }}
  }

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

LCM Import (v2)

The LCM Import (v2) REST API initiates import of a Migration snapshot so that the contents of the application snapshot are imported into the application. You can complete these tasks for imported users: set a specific password for all users in the snapshot, set a unique temporary password for each user in the snapshot, and force password reset at first login.

The presence of status -1 in the response indicates that the import is in progress. You should use the job status URI to determine whether the import is complete.

If the Job completes with status 1, the task details will be mentioned in the items from which the source, destination, and URL to fetch the first set of errors is available. All issues for a particular task can be fetched in the manner of pagination. Acceptable values for `msgtype` are: `error/warn/info`; `limit` represents the number of issues requested per request, and `offset` marks the beginning number to fetch the issues.

This API is version v2.

Required Roles

Service Administrator

Identity Domain Administrator role is required to import user and predefined roles.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-79 LCM Import

Task	Request	REST Resource
LCM Import	POST	/interop/rest/v2/snapshots/import
Import Status	GET	/interop/rest/v2/status/migration/22
Details	GET	/interop/rest/v2/status/migration/22/0/details? limit=25&offset=0&msgtype=info

REST Resource

POST /interop/rest/v2/snapshots/import

Support Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-80 Parameters

Name	Description	Type	Required	Default
snapshotName	Application snapshot that is to be imported	Payload	Yes	None
importUsers	Whether to import users; true imports users and their predefined role assignments. The import fails if the user issuing the request does not have the Identity Domain Administrator role.	Payload	No	false
userPassword	The default password for the imported users.	Payload	No	A unique temporary password is assigned to each user
resetPassword	Whether to force reset password for the imported users on first login, true or false	Payload	No	true

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
snapshots/import
```

```
{
  "snapshotName": "Artifact Snapshot",
  "parameters": {
    "importUsers": true,
```

```

        "userPassword": "epm_cloud",
        "resetPassword": false
    }
}

```

Response

Supported Media Types: application/json

Table 8-81 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be <code>self</code> and/or <code>Job Status</code> . If set to <code>Job Status</code> , you can use the <code>href</code> to get the status of the import operation
data	Parameters as key value pairs passed in the request
items	Details about the resource
source	From where the navigation is being performed
destination	To where the navigation is being performed
name	Name of the task, usually "Task Information"
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body

Response 1 example when import is in progress:

```

{
  "details": null,
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/snapshots/
import",
      "action": "POST",
      "rel": "self",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/24",
      "action": "POST",
      "rel": "Job Status",
      "data": null
    }
  ]
}

```

```
    ]
  }
}
```

Response 2 example when import completes:

```
{
  "details": null,
  "status": 0,
  "items": [
    {
      "name": "Task Information",
      "source": "/Artifact Snapshot/HSS-Shared Services",
      "destination": "Shared Services",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/24/0/details?limit=25&offset=0&msgtype=warning",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    },
    {
      "name": "Task Information",
      "source": "/Artifact Snapshot/HP-Vision",
      "destination": "Vision",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/24/1/details?limit=25&offset=0&msgtype=warning",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    },
    {
      "name": "Task Information",
      "source": "/Artifact Snapshot/DOCREP-Document Repository",
      "destination": "Document Repository",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/migration/24/2/details?limit=25&offset=0&msgtype=warning",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    }
  ],
}
```

```

    {
      "name": "Task Information",
      "source": "/Artifact Snapshot/CALC-Calculation Manager",
      "destination": "Calculation Manager",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/24/3/details?limit=25&offset=0&msgtype=warning",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    },
    {
      "name": "Task Information",
      "source": "/Artifact Snapshot/FDMEE-FDM Enterprise Edition",
      "destination": "FDM Enterprise Edition",
      "links": [
        {
          "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/24/4/details?limit=25&offset=0&msgtype=warning",
          "action": "GET",
          "rel": "Job Details",
          "data": null
        }
      ]
    }
  ],
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
migration/24",
      "action": "GET",
      "rel": "self",
      "data": null
    }
  ]
}

```

Sample cURL command

```

curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H 'Content-Type: application/json' -d '{"snapshotName":"Artifact Snapshot",
"parameters":{"importUsers":true,"userPassword":"epm_cloud",
"resetPassword":false}}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/snapshots/
import'

```

Manage Permission for Manual Access to Database (v1)

Use this API (v1) to manage permission for manual access to database by Oracle.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

This gives you the ability to allow or disallow Oracle personnel to manually access your database in emergency situations when an environment is unresponsive and you have not yet created a service request to investigate the issue.

In an emergency situation, Oracle follows an internal process whereby a high-level development executive, after an independent verification process, permits manual access to the database without your explicit approval. You can also prohibit Oracle from manually accessing the EPM Cloud database, even if a service request to remedy a database issue is open.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v1.

Required Roles

Service Administrator

REST Resource

```
PUT /interop/rest/{api_version}/services/dataaccess?accessType={allow|
revoke}&disableEmergencyAccess={true|false}
```

Parameters:

The following table summarizes the request parameters.

Table 8-82 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
accessType	Allow or revoke the manual access to database by Oracle, where the value can be <code>allow</code> or <code>revoke</code>	Query	Yes	None
disableEmergencyAccess	Whether to prohibit all manual access to the database. The possible values are <code>true</code> and <code>false</code> . Setting this parameter value to <code>true</code> stops Oracle from manually accessing the database even if a service request is open. Oracle does not recommend setting this parameter value to <code>true</code> because Oracle cannot access the database if access is required to troubleshoot and fix a down environment. If the environment is down, you will not be able to issue this REST API to allow Oracle to manually access the database.	Query	No	False

Response

Supported Media Types: application/json

Table 8-83 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

Copy

```
{
  "links": [1]
    0: {
      "rel": "self",      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
dataaccess?accessType=allow&disableEmergencyAccess=true"
      "data": "null",
      "action": "PUT,
    }-
  -
  "details": "null",
  "status": "0"
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Manage Permission for Manual Access to Database (v2)

Use this REST API (v2) to manage permission for manual access to database by Oracle.

This gives you the ability to allow or disallow Oracle personnel to manually access your database in emergency situations when an environment is unresponsive and you have not yet created a service request to investigate the issue.

In an emergency situation, Oracle follows an internal process whereby a high-level development executive, after an independent verification process, permits manual access to

the database without your explicit approval. You can also prohibit Oracle from manually accessing the EPM Cloud database, even if a service request to remedy a database issue is open.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v2

Required Roles

Service Administrator

REST Resource

PUT /interop/rest/v2/services/setmanualdataaccess

Parameters:

The following table summarizes the request parameters.

Table 8-84 Parameters

Name	Description	Type	Required	Default
accessType	Allow or revoke the manual access to database by Oracle, where the value can be allow or revoke	Payload	Yes	None
parameters				
disableEmergencyAccess	Whether to prohibit all manual access to the database. The possible values are true and false. Setting this parameter value to true stops Oracle from manually accessing the database even if a service request is open. Oracle does not recommend setting this parameter value to true because Oracle cannot access the database if access is required to troubleshoot and fix a down environment. If the environment is down, you will not be able to issue this REST API to allow Oracle to manually access the database.	Payload	No	False

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
services/setmanualdataaccess
```

```
{
  "accessType": "Revoke",
  "parameters": {
    "disableEmergencyAccess": "False"
  }
}
```

Response

Supported Media Types: application/json

Table 8-85 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": 0,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/
setmanualdataaccess",
      "action": "PUT",
      "rel": "self",
      "data": null
    }
  ]
}
```

Common Functions

Sample cURL command

```
curl -X PUT -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt -
H 'Content-Type: application/json' -d '{"accessType":"Revoke","parameters":
{"disableEmergencyAccess":"True"}}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/
setmanualdataaccess'
```

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Set Encryption Key (v1)

Provides a Bring Your Own Key (BYOK) solution to include Oracle EPM Cloud in your standard key management rotation.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

The API can be used to set and remove a user-defined encryption key for access to database used in an Oracle EPM Cloud instance.

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-86 Set Encryption Key

Task	Request	REST Resource
Execute a Job	PUT	/interop/rest/ {api_version}/services/ encryptionkey
Retrieve Job Status	GET	/interop/rest/v1/ services/jobs/{jobID}

Parameters:

The following table summarizes the request parameters.

Table 8-87 Parameters

Name	Description	Type	Required	Default
Key	The user-defined encryption key. If empty or no key is passed, encryption is reset.	Form	No	None

Response

Supported Media Types: application/json

Table 8-88 Parameters

Parameters	Description
details	In case of errors, details are published with the error string

Table 8-88 (Cont.) Parameters

Parameters	Description
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "details":null,
  "status":-1,
  "links":[{"
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
encryptionkey",
    "rel":"self",
    "data":null,
    "action":"PUT"
  },{
    "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
jobs/777",
    "rel":"Job Status",
    "data":null,
    "action":"GET"
  }]
}
```

Sample Code**Example 8-35 cURL Sample – setencryptionkey.sh**

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
funcSetEncryptionKey() {
  url=$SERVER_URL/interop/rest/v1/services/encryptionkey

  key="xcfddrerghgArfh"      #use a strong key to set the encryption
key
  param="key=$key"

  funcExecuteRequest "PUT" $url $param "application/json"
```

```

        output=`cat response.txt`
        status=`echo $output | jq '.status'`
        if [ $status == -1 ]; then
            echo "Setting Encryption Key"
            funcGetStatus "GET"
        else
            error=`echo $output | jq '.details'`
            echo "Error occurred. " $error
        fi
        funcRemoveTempFiles "respHeader.txt" "response.txt"
    }

funcResetEncryptionKey() {
    url=$SERVER_URL/interop/rest/v1/services/encryptionkey

    param=""

    funcExecuteRequest "PUT" $url $param "application/json"
    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Resetting Encryption Key"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Set Encryption Key (v2)

The Set Encryption Key (v2) REST API provides a Bring Your Own Key (BYOK) solution to include Oracle EPM Cloud in your standard key management rotation.

The API can be used to set and remove a user-defined encryption key for access to database used in an Oracle EPM Cloud instance.

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 8-89 Set Encryption Key

Task	Request	REST Resource
Execute a Job	PUT	/interop/rest/v2/services/setencryptionkey
Retrieve Job Status	GET	/interop/rest/v2/status/jobs/777

REST Resource

PUT /interop/rest/v2/services/setencryptionkey

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-90 Parameters

Name	Description	Type	Required	Default
parameters				
key	The user-defined encryption key. If empty or no key is passed, encryption is reset.	Payload	No	None

Example URL and Payload

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/setencryptionkey

```
{"parameters": {"key": "se!m+a2J"}}
```

Response

Supported Media Types: application/json

Table 8-91 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
services/setencryptionkey",
    "data": null,
    "action": "PUT"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/jobs/8921378039543483",
    "data": null,
    "action": "GET"
  }],
  "details": null
}
```

Sample cURL command

```
curl -X PUT -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d '{"parameters":
{"key": "abc"}}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
services/setencryptionkey'
```

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Migration Status Codes

The status code returned in the response of REST API calls identifies the status of the operation.

Table 8-92 Status Code

Status Code	Description
0	Operation success
-1	Operation in progress
+ve	Operation failed, with the status signifying an error

Copy a File Between Instances (v1)

Use this API (v1) to copy a file from one environment (source) to another environment (target).

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

This API is executed on the target environment after details are provided for the source environment from which the file is to be copied. This feature gives you flexibility in copying files from one cloud environment to another.

Prerequisites: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the Copy File REST API. See the `encrypt` command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v1/services/copyfile`

Supported Media Types: `application/x-www-form-urlencoded`

Parameters:

The following table summarizes the POST request parameters.

Table 8-93 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version, such as v1	Path	Yes	None
<code>sourceFileName</code>	Name of the file to be copied	Form	Yes	None
<code>userName</code>	User with access to the source instance	Form	Yes	None
<code>pwd</code>	The location and name of the file containing the encrypted password for the user. The encrypted password must then be passed as one of the parameters for the Copy File REST API. For information on encrypting and generating the <code>password.epw</code> file with EPM Automate, see the <code>encrypt</code> command in Command Reference in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i> .	Form	Yes	None
<code>sourceURL</code>	The URL of the source instance	Form	Yes	None

Table 8-93 (Cont.) Parameters

Name	Description	Type	Required	Default
targetFileName	Name of the file to be copied to the target environment	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 8-94 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": -1,
  "items": null,
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/copyfile",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/jobs/1502357937045",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Copy a File Between Instances (v2)

Use this REST API (v2) to copy a file from one environment (source) to another environment (target).

This API is executed on the target environment after details are provided for the source environment from which the file is to be copied. This feature gives you flexibility in copying files from one cloud environment to another.

Prerequisites: The password of the source EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password must then be passed as one of the parameters for the Copy File REST API. See the `encrypt` command in [Command Reference](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/v2/files/copyfrominstance`

Supported Media Types: `application/json`

Parameters:

The following table summarizes the POST request parameters.

Table 8-95 Parameters

Name	Description	Type	Required	Default
<code>sourceFileName</code>	Name of the file to be copied	Payload	Yes	None
<code>userName</code>	User with the Service Administrator predefined role in the source instance	Payload	Yes	None
<code>pwd</code>	The location and name of the file containing the encrypted password for the user. For information on encrypting and generating the <code>password.epw</code> file with EPM Automate, see the <code>encrypt</code> command in Command Reference in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i> .	Payload	Yes	None
<code>sourceURL</code>	The URL of the source instance	Payload	Yes	None
<code>targetFileName</code>	Name of the file to be copied to the target environment	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/files/
copyfrominstance
```

```
{
  "sourceFileName": "<NAME>",
  "userName": "<USERNAME>",
  "pwd": "<PASSWORDFILE>",
  "sourceURL": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com",
  "targetFileName": "<NAME>"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-96 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
"status": -1,
"items": null,
"links": [{
  "rel": "self",
  "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
files/copyfrominstance",
  "data": null,
  "action": "POST"
}, {
  "rel": "Job Status",
  "href": "https://<SERVICE_NAME>-
```

```
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/  
jobs/1502357937045",  
    "data": null,  
    "action": "GET"  
  }},  
  "details": null  
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt  
-H 'Content-Type: application/json' -d  
'{"sourceFileName":"FILE_TO_BE_COPIED",  
"sourceURL":"SOURCE_URL","userName":"USER_NAME","targetFileName":"TARGET_FILE  
NAME","pwd":"ENCRYPTED_PASSWORD"}' 'https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/files/  
copyfrominstance'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Skip Updates (v1)

Use the Skip Updates (v1) API to add, list, or remove a skip update request. Using this API, you can ask Oracle to skip applying a monthly update to an environment, or remove all previous skip update requests so that the environment is updated to the main code line. This allows you to skip updates to an EPM Cloud production environment at times when you need to complete time-sensitive tasks, for example, closing a quarter, without creating a service request.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

You can also list the skip update requests currently specified for an environment. You can set skip update requests for a maximum of two update cycles. You cannot skip updates for an environment that is on a one-off patch. Additionally, you cannot skip monthly updates that are more than two months apart from the update that the environment is currently on. For example, if the environment is currently on 20.12, you can skip 21.01 and 21.02 updates, but not 21.03. This gives you better control over monthly and weekly updates.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v1/services/skipupdate

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-97 Parameters

Name	Description	Type	Required	Default
type	The desired action to be performed (Add, List, or Remove)	Form	Yes	None
version	The version to be skipped from update, for example, 20.12	Form	No (Yes, only when the type is Add)	None
comment	The business reason for skipping an update, such as Year End	Form	No (Yes, only when the type is Add)	None

Response

Supported Media Types: application/json

Parameters:

Table 8-98 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [{
```

```
"https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v1/services/skipupdate",
    "rel":"self",
    "data":null,
    "action":"POST"
  }
],
"details":null,
"status":0
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Skip Updates (v2)

Use the Skip Updates (v2) REST API to add, list, or remove a skip update request. Using this API, you can ask Oracle to skip applying a monthly update to an environment, or remove all previous skip update requests so that the environment is updated to the main code line. This allows you to skip updates to an EPM Cloud production environment at times when you need to complete time-sensitive tasks, for example, closing a quarter, without creating a service request.

You can also list the skip update requests currently specified for an environment. You can set skip update requests for a maximum of two update cycles. You cannot skip updates for an environment that is on a one-off patch. Additionally, you cannot skip monthly updates that are more than two months apart from the update that the environment is currently on. For example, if the environment is currently on 22.10, you can skip 22.11 and 22.12 updates, but not 23.01. This gives you better control over monthly and weekly updates.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v2/services/skipupdate

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-99 Parameters

Name	Description	Type	Required	Default
type	The desired action to be performed (Add, List, or Remove)	Payload	Yes	None
parameters version	The version to be skipped from update, for example, 22.12	Payload	No (Yes, only when the type is Add)	None
comment	The business reason for skipping an update, such as Year End	Payload	No (Yes, only when the type is Add)	None

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
services/skipupdate
```

```
{
  "type": "Add",
  "parameters": {
    "version": "22.12",
    "comment": "Year end"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-100 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status

Table 8-100 (Cont.) Parameters

Name	Description
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "status": 0,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/
skipupdate",
    "data": null,
    "action": "POST"
  }
  "details": null
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H 'Content-Type: application/json' -d '{"type":"Add","parameters":
{"version":"22.10","comment":"Year End"}}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/services/
skipupdate'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Send Email (v1)

Use the Send Mail (v1) REST API to send an email to specified recipients, optionally attaching files from EPM Cloud. You can attach any file up to 10 MB in size, other than a snapshot, that is available in EPM Cloud environments. This API can be incorporated into REST API programs and scripts to notify users of various conditions or to send reports.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the process is in progress. Any non-zero status except -1 indicates failure.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

Table 8-101 Tasks

Task	Request	REST Resource
Trigger sendmail	POST	/interop/rest/<api_version>/services/sendmail
Retrieve sendmail status	GET	/interop/rest/<api_version>/services/jobs/777

REST Resource

POST /interop/rest/<api_version>/services/sendmail

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 8-102 Parameters

Name	Description	Type	Required	Default
<i>api_version</i>	Specific API version	Path	Yes	None
to	The recipient email addresses separated by semi-colons	Form	Yes	None
subject	The subject for the email	Form	Yes	None
body	The body of the email providing details.	Form	Yes	None
attachments	One or more file names separated by commas to be added as attachments to the email, for example, outbox/Errorfile.txt or inbox/Errorfile2.txt You can attach any file up to 10 MB in size, other than a snapshot, that is available in EPM Cloud environments.	Form	No	None

Sample request payload

```
to:abc@oracle.com
subject:EPM
```

```
body:EPM weekly email
attachments:apr/2021-08-10 11_30_25/2021-08-10 11_30_25.html
```

Response

Parameters:

Table 8-103 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body:

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
sendmail",
    "data": null,
    "action": "POST"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
jobs/1502357937045",
    "data": null,
    "action": "GET"
  }],
  "details": null
}
```

JobID is appended only to the API used to fetch the status of the progress.

Sample code:

```
public void sendMail() throws Exception {
```



```

String to = "RECIPIENT EMAIL ADDRESS";
String subject = "SUBJECT OF THE MAIL";
String body = "BODY OF THE MAIL";
String attachments = "NAME OF THE FILE TO BE ATTACHED";

String urlString = String.format("%s/interop/rest/v1/services/
sendmail", serverUrl);

String params = "to=" + to + "&subject=" + subject + "&body=" +
body + "&attachments=" + attachments;

String response = executeRequest(urlString, "POST", params,
"application/x-www-form-urlencoded");

getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"GET");
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Send Email (v2)

Use the Send Mail (v2) REST API to send an email to specified recipients, optionally attaching files from EPM Cloud. You can attach any file up to 10 MB in size, other than a snapshot, that is available in EPM Cloud environments. This API can be incorporated into REST API programs and scripts to notify users of various conditions or to send reports.

This topic describes the simplified v2 version of this REST API. This version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. This API is backwards compatible.

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the process is in progress. Any non-zero status except -1 indicates failure.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v2.

Required Roles

Service Administrator

Table 8-104 Tasks

Task	Request	REST Resource
Trigger sendmail	POST	/interop/rest/v2/emails/send

Table 8-104 (Cont.) Tasks

Task	Request	REST Resource
Retrieve sendmail status	GET	/interop/rest/v2/status/jobs/777

REST Resource

POST /interop/rest/v2/emails/send

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-105 Parameters

Name	Description	Type	Required	Default
to	The recipient email addresses separated by semi-colons	Payload	Yes	None
subject	The subject for the email	Payload	Yes	None
body	The body of the email providing details.	Payload	Yes	None
parameters	Parameters for this REST API	Payload	No	None
attachments	One or more file names separated by commas to be added as attachments to the email, for example, outbox/Errorfile.txt or inbox/Errorfile2.txt You can attach any file up to 10 MB in size, other than a snapshot, that is available in EPM Cloud environments.	Payload	No	None

Sample URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/emails/send
```

```
{
  "to": "<EMAIL_ADDRESS>",
  "subject": "EPM",
  "body": "EPM weekly email",
  "parameters": {
    "attachments": "apr/Feedback 2022-03-01 07_42_04/access_log.zip"
  }
}
```

Response

Parameters:

Table 8-106 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; <code>rel</code> is "Job Details"

Example of Response Body:

```
{
  "status": -1,
  "items": null,
  "links": [{
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
mails/send",
    "data": null,
    "action": "POST"
  }, {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/jobs/1502357937045",
    "data": null,
    "action": "GET"
  }],
  "details": null
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' -d
'{"subject": "SUBJECT_OF_THE_MAIL",
"to": "RECIPIENT_EMAIL_ADDRESS", "body": "BODY_OF_THE_MAIL", "parameters":
{"attachments": "NAME_OF_THE_FILE_TO_BE_ATTACHED"}}' 'https://
```

```
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/
rest/v2/emails/send'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Clone an Environment

Use this REST API to clone the current environment and, optionally, identify domain artifacts (users and predefined roles), Data Management records, audit records, Job Console records, contents of the inbox and outbox, and stored snapshots.

This API is executed on the source environment after details are provided for the target environment to be cloned. It is an alternative to using the Clone Environment feature in a browser or the EPM Automate cloneEnvironment command.

Prerequisites: The password of the target EPM Cloud environment must have already been encrypted using EPM Automate. The encrypted password string must then be passed as one of the parameters for the Clone Environment REST API. See the [encrypt command](#) in *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

For considerations with cloning an environment, see [Cloning EPM Cloud Environments](#) in *Administering Migration for Oracle Enterprise Performance Management Cloud*.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This REST API is version v1.

Required Roles

Service Administrator

Identity Domain Administrator role is required to clone users and predefined roles.

REST Resource

POST /interop/rest/v1/services/clone

Supported Media Types: application/json

Parameters:

The following table summarizes the POST request parameters.

Table 8-107 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version, such as v1	Path	Yes	None

Table 8-107 (Cont.) Parameters

Name	Description	Type	Required	Default
targetUsername	The ID of a Service Administrator in the target environment. If you plan to clone user and role assignments in the target environment, this user must also have the Identity Domain Administrator role.	Payload	Yes	None
targetEncryptPassword	The encrypted password of the user identified by targetUsername to be passed as a string. For information on encrypting and generating the password.epw file with EPM Automate, see encrypt in <i>Working with EPM Automate for Oracle Enterprise Performance Management Cloud</i> .	Payload	Yes	None
targetURL	The URL of the environment that will become the cloned environment	Payload	Yes	None
snapshotName	Optionally, the name of a snapshot that should be used for cloning. This snapshot must be present in the source environment.	Payload	No	Last maintenance snapshot
migrateUsers	Whether to clone users and their predefined and application role assignments, True or False. For this option to work, the user identified by targetUsername must have the Identity Domain Administrator role in the target environment.	Payload	No	False
maintenanceStartTime	Whether to reset the maintenance start time of the cloned environment to that of the source environment. To keep the current maintenance start time of the target environment, set this value to False.	Payload	No	True
dataManagement	Whether to clone the Data Management records of the source to the target environment for all environments other than Oracle Enterprise Data Management Cloud and Narrative Reporting, True or False.	Payload	No	True
applicationAudit	Whether to clone the application audit data in the source to the target environment for FreeForm, Planning, Planning Modules, Financial Consolidation and Close, and Tax Reporting environments, True or False. Including application audit data while cloning an environment makes the target environment keep the application audit history.	Payload	No	True
jobConsole	Whether to clone the job console data in the source to the target environment for FreeForm, Planning, Planning Modules, Financial Consolidation and Close, and Tax Reporting environments, True or False. Including job console data while cloning an environment makes the target environment keep the job console history.	Payload	No	True

Table 8-107 (Cont.) Parameters

Name	Description	Type	Required	Default
storedSnapshotsAndFiles	Whether to clone the contents of the inbox, outbox, and stored snapshots in the source to the target environment for all environments, True or False. Including stored snapshots and files while cloning an environment makes sure that the stored snapshots and files are not lost. This is useful especially while cloning a Classic environment to an OCI (Gen 2) environment.	Payload	No	False

Example URL and Payload

```
{
  "targetURL": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com ",
  "targetUserName": "cloneUser@oracle.com",
  "targetEncryptPassword": "<targetUserEncryptedPasswordString>",
  "parameters": {"snapshotName": "Artifact Snapshot"
  "migrateUsers": "true", "maintenanceStartTime": "true",
  "dataManagement": "true", "jobConsole": "true", "applicationAudit": "true",
  "storedSnapshotsAndFiles": "false"}
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-108 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
intermittentStatus	Status of each step performed; can be polled regularly from the job status URL

Example of Response Body

The following shows an example of the response body in JSON format.

```
{"intermittentStatus":null, "links":[{"rel":"Job Status","href":"https://
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/
```

```
rest/v1/services/
status", "data":null, "action":"GET"}], "details":null, "status":-1"items":
null)
```

Example 8-36 Java Sample – CloneEnvironment.java

Prerequisites: json.jar

Common functions: See [Appendix A, Common Helper Functions for Java](#).

```
//
// BEGIN - Clone Environment
//
public void cloneEnvironment() throws Exception {
    String targetUrl = "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com" ;
    String targetUsername = "<Target User name>";
    String encryptedPwdString =
fetchPwdFromFile("epw_file_Path") ; //"<Target system encrypted
password>";
    String snapshotToClone = "Artifact Snapshot";
    JSONObject params = new JSONObject();
    JSONObject payload = new JSONObject();
    //Optional parameters if not passed default values will be
picked
    /*
    params.put("snapshotName", snapshotToClone);
    params.put("migrateUsers", Boolean.TRUE.toString());
    params.put("maintenanceStartTime", Boolean.TRUE.toString());
    params.put("dataManagement", Boolean.TRUE.toString());
    */
    //Mandatory parameters
    payload.put("targetURL", targetUrl);
    payload.put("targetUserName", targetUsername);
    payload.put("targetEncryptPassword", encryptedPwdString);
    payload.put("parameters", params);

    String urlString = String.format("%s/interop/rest/v1/services/
clone", serverUrl);
    String response = executeRequest(urlString, "POST",
payload.toString(), "application/json");

    getMigrationJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
}

private String fetchPwdFromFile(String filePath) {
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader(filePath));
        String line = null;
        String pwdString = null;
        while ((line = br.readLine()) != null) {
            pwdString = line;
        }
    }
}
```

```

    }
    return pwdString;
} catch (Exception e) {
} finally {
    if (null != br)
        try {
            br.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
}

return null;
}
//
// END - Clone Environment
//

```

Example 8-37 cURL Sample – cloneEnvironment.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

funcCloneEnvironment() {
    url="$SERVER_URL/interop/rest/v1/services/clone"
    local targetUrl="<<TargetSystemURL>"
    local targetUserName="<<TargetSystemUsername>"
    local targetEncryptedPassword= $(cat $EPWfilePath)
    #"<TargetSystemEncryptedPasswordString>"

    #optionalParams="{\"snapshotName\": \"Artifact
Snapshot\", \"migrateUsers\": \"true\", \"maintenanceStartTime\": \"true\", \"data
Management\": \"true\"}"

    param="{\"targetURL\": \"$targetUrl\", \"targetUserName\": \"$targetUserName\", \
\"targetEncryptPassword\": \"$targetEncryptedPassword\"}"
    #, \"parameters\": \"$optionalParams\"}"
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=$(cat response.txt)
    status=$(echo $output | jq '.status')
    if [ $status == -1 ]; then
        echo "CloneEnvironment is in progress.."
        funcGetStatus "GET"
    else
        error=$(echo $output | jq '.details')
        echo "Error occurred. " $error
    fi
}

```


Example 8-38 Groovy Sample – cloneEnvironment.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def
cloneEnvironment(targetURL,targetUsername,targetEncryptedPasswordFile){
    String scenario = "Clone Environment";
    def targetEncryptedPassword =
fetchPwdFromFile(targetEncryptedPasswordFile);
    def json = new JsonBuilder()
    //Optional parameter to be set if needed
    //def optionalParams = [snapshotName: "Artifact Snapshot",
migrateUsers: "true", maintenanceStartTime:
"true" ,dataManagement:"true"]

    def payload = new JsonBuilder()
        payload targetURL: targetURL,
            targetUserName: targetUsername,
            targetEncryptPassword: targetEncryptedPassword //,
            //parameters: optionalParams

    params=payload.toString();
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/v1/services/clone");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/json");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}
def fetchPwdFromFile(filePath) {
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader(filePath));
        String line = null;
        String pwdString = null;
        while ((line = br.readLine()) != null) {
            pwdString = line;
        }
        return pwdString;
    } catch (Exception e) {
    } finally {
        if (null != br)
            try {
                br.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
            }
    }
}
```

```

        e.printStackTrace();
    }

}

return null;
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Export Essbase Data (v2)

The Export Essbase (v2) REST API exports level 0 or all data for the specified cube. Export data files are written to the Outbox directory as a zip file. You can download it using the EPM Automate [downloadFile](#) command or the [Download](#) REST API.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /interop/rest/v2/essbase/export

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 8-109 Parameters

Name	Description	Type	Required	Default
cubeName	Name of the BSO or ASO cube	Payload	Yes	None
fileName	Name of a zip file in which the Essbase data will be exported. This file will be available in the outbox from where it can be downloaded.	Payload	Yes	None
level	The value is 0 or ALL, Level 0 and ALL are valid for BSO cubes. Only level 0 is valid for ASO cubes.	Payload	Yes	level=0

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
essbase/export
```

```
{
  "cubeName": "Plan1",
  "fileName": "Plan1Export.zip",
  "parameters": {
    "level": "0"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 8-110 Parameters

Attribute	Description
details	In case of errors, details are published with the error string
status	See Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values. Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the re-export operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following are examples of the response body in JSON format.

Response 1 example where the export is in progress:

```
{
  "details": "Essbase Database Export",
  "status": -1,
  "items": null,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
essbase/export",
      "action": "POST",
```

```

        "rel": "self",
        "data": null
    },
    {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
jobs/19974850954170405",
        "action": "GET",
        "rel": "Job Status",
        "data": null
    }
]
}

```

Response 2 example when export successfully completes:

```

{
    "details": null,
    "status": 0,
    "items": null,
    "links": [
        {
            "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
jobs/19974850954170405",
            "action": "GET",
            "rel": "self",
            "data": null
        }
    ]
}

```

Sample cURL command

```

curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H
'Content-Type: application/json' -d '{"cubeName": "Plan1","fileName":
"Plan1Export.zip","parameters":{"level": "0" }}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/essbase/
export'

```

9

Viewing and Setting the Daily Maintenance Window Time

This section describes the REST APIs for the daily maintenance window time.

Getting and Setting the Daily Maintenance Window Time

Use these REST APIs to get the current build version and daily maintenance window time, and to set the daily maintenance window time. You can also run the daily maintenance while skipping the scheduled daily maintenance.

This API is version v1.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 9-1 Getting and Setting the Daily Maintenance Time

Task	Request	REST Resource
Get the Build Version and Daily Maintenance Window Time	GET	/interop/rest/ {api_version}/services/ dailymaintenance
Get the Build Version and Daily Maintenance Window Time (v2)	GET	/interop/rest/v2/ maintenance/ getdailymaintenancestartt ime
Set The Daily Maintenance Window Time	PUT	/interop/rest/ {api_version}/services/ dailymaintenance? StartTime={N}
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v1)	POST	/interop/rest/ {api_version}/services/ maintenancewindow
Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)	POST	/interop/rest/v2/ maintenance/ rundailymaintenance

Get the Build Version and Daily Maintenance Time (v1)

This API (v1) returns information about the current build version and the scheduled daily maintenance window time.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not

require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

This API is version v1.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrator

REST Resource

GET /interop/rest/{api_version}/services/dailymaintenance

Parameters:

The following table summarizes the request parameters.

Table 9-2 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None

Response

Supported Media Types: application/json

Table 9-3 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
items	Detailed information about the API
amwTime	Scheduled Time (in HH:00 format using a 24 hour clock) at which the maintenance process should start and an optional time zone. Acceptable start time value range is 00:00 - 23:00. If the start time is not to be set in UTC, specify a valid standard time zone; for example, "14:00 America/Los_Angeles" for 2:00 pm Pacific Standard Time.
buildVersion	Current build version
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details":null,
  "links":[
    {
      "rel":"self","href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/services/
dailymaintenance",
      "data":"null",
      "action":"GET"}],
      "status":"0",
      "items":[
        {
          "amwTime":"19",
          "buildVersion":"16.10.17"}
      ]
    }
  }
```

Get Build Version and Daily Maintenance Time Sample Code

Example 9-1 Java Sample – GetMaintenanceDetails.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN
//
public void getMaintenanceDetails () throws Exception {
    String urlString = String.format("%s/interop/rest/v1/services/
dailymaintenance", serverUrl);
    String response = executeRequest(urlString, "GET", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        JSONArray fileList = json.getJSONArray("items");
        System.out.println("List of items are :");
        JSONObject jsonObj = null;
        for(int i=0; i<fileList.length(); i++){
            jsonObj = (JSONObject)fileList.get(i);
            System.out.println("build version : " +
jObj.getString("buildVersion"));
            System.out.println("AMW time : " + jsonObj.getString("amwTime"));
            System.out.println("Link : " + ((JSONObject)
((JSONArray)json.getJSONArray("links")).get(0)).getString("href") + "\n");
        }
    }
}
//
// END
//
```

Example 9-2 cURL Sample – GetMaintenanceDetails.shPrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [Common Helper Functions for cURL](#)

```

funcGetMaintenanceDetails () {
    url=$SERVER_URL/interop/rest/v1/services/dailymaintenance
    funcExecuteRequest "GET" $url

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "List of items :"
        count=`echo $output | jq '.items | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Build Version : " `echo $output | jq
'.items['$i'].buildVersion'`
            echo "AMW Time : " `echo $output | jq '.items['$i'].amwTime`
            echo "Link : " `echo $output | jq '.links[0].href'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 9-3 Groovy Sample – GetMaintenanceDetails.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def getMaintenanceDetails () {
    def url;
    try {
        url = new URL(serverUrl + "/interop/rest/v1/services/
dailymaintenance ")
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "GET", null);
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == 0 ) {
        def items = object.items
        println "List of items :"
        items.each{
            println "Build Version : " + it.buildVersion
            println "AMW Time : " + it.amwTime
            println "Link : " + it.links[0].href + "\n"
        }
    }
}

```



```

    }
  } else {
    println "Error occurred while listing versions"
    if (object.details != null)
      println "Error details: " + object.details
  }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Get the Build Version and Daily Maintenance Window Time (v2)

This REST API (v2) returns information about the current build version and the scheduled daily maintenance window start time.

This API is version v2

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrator

REST Resource

GET /interop/rest/v2/maintenance/getdaily maintenancestarttime

Parameters:

The following table summarizes the request parameters.

Table 9-4 Parameters

Name	Description	Type	Required	Default
showTimeZone	Displays the time in the time zone specified while setting it	Query	No	false

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/maintenance/getdaily maintenancestarttime?showTimeZone=true
```

Response

Supported Media Types: application/json

Table 9-5 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
items	Detailed information about the API
amwTime	Scheduled start time for the daily maintenance window in 24 hour format. The time is provided with respect to the time zone specified when setting it.
buildVersion	Current build version
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": 0,
  "items": [
    {
      "buildVersion": "22.09.40",
      "amwTime": "19:00",
      "timeZone": "Etc/UTC"
    }
  ],
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
maintenance/getdaily maintenancestarttime",
      "action": "GET",
      "rel": "self",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X GET -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H 'Content-Type: application/json' 'https://
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/maintenance/getdaily maintenancestarttime?
showTimeZone=true'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Setting the Daily Maintenance Time (v1)

Use this REST API (v1) to set the daily maintenance window start time.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

Use this REST API to set the daily maintenance window time.

Note: To ensure that the use of this API does not interfere with the Oracle requirement for creating backups, this API will not change the maintenance start time if the daily maintenance process did not run in the last 36 hours.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v1.

Required Roles

Service Administrator

REST Resource

PUT `/interop/rest/{api_version}/services/dailymaintenance?StartTime={N}`

Parameters:

The following table summarizes the request parameters.

Table 9-6 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
StartTime	Scheduled Time (in HH:00 format using a 24 hour clock) at which the maintenance process should start and an optional time zone. Acceptable start time value range is 00:00 - 23:00. If the start time is not to be set in UTC, specify a valid standard time zone; for example, "14:00 America/Los_Angeles" for 2:00 pm Pacific Standard Time.	Query	Yes	None

Response

Supported Media Types: `application/json`

Table 9-7 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [1]
  0: {
    "rel": "self",
    "href": "https://<SERVICE_NAME>-
    <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
    services/dailymaintenance?StartTime=23"
    "data": "null",
    "action": "PUT",
  }-
  -
  "details": "null",
  "status": "0"
}
```

Maintenance Window Time Sample Code**Example 9-4 Java Sample – SetMaintenanceDetails.java**

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN
//
public void setMaintenanceDetails () throws Exception {
    String urlString = String.format("%s/interop/rest/v1/services/
dailymaintenance?StartTime=23", serverUrl);
    String response = executeRequest(urlString, "PUT", null);
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        System.out.println("Updated successfully");
    }
}
}
```

```
//  
// END  
//
```

Example 9-5 cURL Sample – SetMaintenanceDetails.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
uncSetMaintenanceDetails () {  
    rl=$SERVER_URL/interop/rest/v1/services/dailymaintenance?StartTime=23  
    funcExecuteRequest "PUT" $url  
  
    output='cat response.txt'  
    status='echo $output | jq '.status''  
    if [ $status == 0 ]; then  
        echo "Updated Successfully"  
    else  
        error='echo $output | jq '.details''  
        echo "Error occurred. " $error  
    fi  
    funcRemoveTempFiles "respHeader.txt" "response.txt"  
}
```

Example 9-6 Groovy Sample – SetMaintenanceDetails.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def setMaintenanceDetails () {  
    def url;  
    try {  
        url = new URL(serverUrl + "/interop/rest/v1/services/  
dailymaintenance?StartTime=23 ")  
    } catch (MalformedURLException e) {  
        println "Malformed URL. Please pass valid URL"  
        System.exit(0);  
    }  
    response = executeRequest(url, "PUT", null);  
    def object = new JsonSlurper().parseText(response)  
    def status = object.status  
    if (status == 0 ) {  
        println "Updated Successfully"  
    } else {  
        println "Error occurred while listing versions"  
        if (object.details != null)  
            println "Error details: " + object.details  
    }  
}
```

Setting the Daily Maintenance Time (v2)

Use this REST API (v2) to set the daily maintenance window start time.

Note: To ensure that the use of this API does not interfere with the Oracle requirement for creating backups, this API will not change the maintenance start time if the daily maintenance process did not run in the last 36 hours.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v2

Required Roles

Service Administrator

REST Resource

PUT /interop/rest/v2/maintenance/setdaily maintenancestarttime

Parameters:

The following table summarizes the request parameters.

Table 9-8 Parameters

Name	Description	Type	Required	Default
StartTime	Scheduled time (in HH:00 format using a 24 hour clock) at which the maintenance process should start and an optional time zone. Acceptable start time value range is 00:00 - 23:00. If the start time is not to be set in UTC, specify a valid standard time zone; for example, "14:00 America/Los_Angeles" for 2:00 pm Pacific Standard Time.	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
maintenance/setdaily maintenancestarttime
```

```
{"startTime":"08:00"}
```

Response

Supported Media Types: application/json

Table 9-9 Parameters

Parameters	Description
details	In case of errors, details are published with the error string

Table 9-9 (Cont.) Parameters

Parameters	Description
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible value: self
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": 0,
  "items": null,
  "links": [
    {
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
maintenance/setdaily maintenancestarttime",
      "action": "PUT",
      "rel": "self",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X PUT -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt -
H 'Content-Type: application/json' -d '{"startTime":"08:00"}' 'https://
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/
rest/v2/maintenance/setdaily maintenancestarttime'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v1)

Use this API (v1) if you want to run daily maintenance while skipping the scheduled daily maintenance.

This topic describes the original version of this REST API. You can also use the [simplified v2 version of the REST API](#). The v2 version contains all parameters in the payload and does not require URL encoding while calling the REST APIs. This makes the v2 API easier to use. The v2 version is backwards compatible.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v1.

Required Roles

Service Administrator

REST Resource

POST `/interop/rest/{api_version}/services/maintenancewindow`

Parameters:

The following table summarizes the request parameters.

Table 9-10 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version	Path	Yes	None
<code>skipNext</code>	Specifies if the set maintenance time should be used, true or false	String	Yes	false

Response

Supported Media Types: `application/json`

Table 9-11 Parameters

Parameters	Description
<code>details</code>	In case of errors, details are published with the error string
<code>status</code>	See Migration Status Codes
<code>links</code>	Detailed information about the link
<code>href</code>	Links to API call
<code>action</code>	The HTTP call type
<code>rel</code>	Possible value: <code>self</code>
<code>data</code>	Parameters as key value pairs passed in the request

- Java Code using `skipNext`:


```
DailyMaintenanceWithSkipNextv1.java Main method:
runDailyMaintenanceWithSkipNext() Helper method waitForCompletion
```

- **Curl Code using skipNext:**
Main method: funcSetMaintenanceWithSkipNext
- **Groovy Code using skipNext:**
DailyMaintenanceWithSkipNextv1.groovy Main method:
runDailyMaintenanceWithSkipNext()

Maintenance Window Time Sample Code

Example 9-7 Java Sample – SetMaintenanceDetails.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
public class DailyMaintenanceWithSkipNextv1 {
    private String userName ; // PBCS user name
    private String password ; // PBCS user password
    private String serverUrl ; // PBCS server URL
    private String lcmVersion = "v1"; // Version of the PBCS API that you are

    public void runDailyMaintenanceWithSkipNext(String comment) throws
Exception {
        Scanner in = null;
        try {

            String skipNext = "false";
            JSONObject params = new JSONObject();

            /*Parameter to Skip the next scheduled maintenance report to be
run .
It is either true or false
If true the scheduled daily maintenance is run
If false the scheduled daily maintenance is skipped*/

            params.put("skipNext", skipNext);

            String urlString = String.format(
                "%s/interop/rest/%s/services/maintenancewindow",
serverUrl,
                lcmVersion);
            String response = executeRequest(urlString, "POST",
                params.toString(), "application/json");
            waitForCompletion(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
        } catch (Exception e) {

        } finally {
            in.close();
        }
    }
}
```

```

        private void waitForCompletion(String pingUrlString, String
methodType)
            throws Exception {
                boolean completed = false;
                while (!completed) {
                    try {
                        String pingResponse = executeRequest(pingUrlString,
methodType,
                            null, "application/x-www-form-urlencoded");
                        JSONObject json = new JSONObject(pingResponse);
                        int status = json.getInt("status");
                        if (status == -1) {
                            try {
                                System.out.println("Please wait...");
                                Thread.sleep(20000);
                            } catch (InterruptedException e) {
                                completed = true;
                                throw e;
                            }
                        }
                        } else {
                            if (status > 0) {
                                System.out.println("Error occurred: "
+ json.getString("details"));
                            } else {
                                System.out.println("Completed");
                            }
                            completed = true;
                        }
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                        // services are down, waiting to come up
                        Thread.sleep(60000);
                    }
                }
            }
    }
}

```

Example 9-8 cURL Sample – SetMaintenanceDetails.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

#!/bin/sh
SERVER_URL=""
USERNAME=""
PASSWORD=""
API_VERSION=""
DOMAIN=""

```

```

funcSetMaintenanceWithSkipNext () {
    url=${SERVER_URL}/interop/rest/v1/services/maintenancewindow

    skipNext="false"

    param="{\"skipNext\": \"${skipNext}\"}"
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Daily Maintenance succesfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 9-9 Groovy Sample – SetMaintenanceDetails.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

class DailyMaintenanceWithSkipNextv1 {

    def userName // PBCS user name
    def password // PBCS user password
    def serverUrl // PBCS server URL
    def lcmVersion = "v1"; // Version of the PBCS API that you are

    void runDailyMaintenanceWithSkipNext() throws Exception {
        def skipNext = "false"; //true or false based on requirement

        JSONObject params = new JSONObject();
        params.put("skipNext", skipNext);

        String urlString = String.format("%s/interop/rest/%s/services/
maintenancewindow", serverUrl, lcmVersion);
        String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
        waitForCompletion(fetchPingUrlFromResponse(response, "Job Status"));
    }
}

```

Running Daily Maintenance While Skipping the Scheduled Daily Maintenance (v2)

Use this REST API (v2) if you want to run daily maintenance with an option to skip the next scheduled daily maintenance.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [About the REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

This API is version v2

Required Roles

Service Administrator

REST Resource

POST /interop/rest/v2/maintenance/rundailymaintenance

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 9-12 Parameters

Name	Description	Type	Required	Default
skipNext	Specifies if the next scheduled daily maintenance time should be skipped, true or false	String	Yes	false

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
maintenance/rundailymaintenance
```

Response

Supported Media Types: application/json

```
{"skipNext":"true"}
```

Table 9-13 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call

Table 9-13 (Cont.) Parameters

Parameters	Description
action	The HTTP call type
rel	Possible value: self
data	Null

Example of the Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": -1,
  "items": null,
  "links": [
    {
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
maintenance/rundailymaintenance",
      "action": "POST",
      "rel": "self",
      "data": null
    },
    {
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/status/
service/maintenancewindow/1660115130723",
      "action": "GET",
      "rel": "Job Status",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D respHeader.txt
-H 'Content-Type: application/json' -d '{"skipNext": "true"}' 'https://
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/
rest/v2/maintenance/rundailymaintenance'
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

10

Managing Users

This section describes the REST APIs to manage users, including these APIs:

- [Adding users to an Identity Domain](#)
- [Removing users from an identity domain](#)
- [Adding users to a predefined role](#)
- [Removing users' role assignment](#)
- [Updating users](#)
- [Adding a user to a batch of groups](#)
- [Removing a user from a batch of groups](#)
- [Adding groups](#)
- [Removing groups](#)
- [User group report](#)
- [User Access Report \(v1\)](#)
- [User Access Report \(v2\)](#)
- [User Audit Report \(v1\)](#)
- [User Audit Report \(v2\)](#)
- [Role assignment report](#)
- [Role Assignment Audit Report for OCI \(Gen 2\) Environments](#)
- [Invalid Login Report for OCI \(Gen 2\) Environments](#)
- [Group Assignment Audit Report](#)
- [Adding users to a team for Account Reconciliation](#)
- [Adding users to a team for Financial Consolidation and Close and Tax Reporting](#)
- [Removing users from a team for Account Reconciliation](#)
- [Removing users from a team for Financial Consolidation and Close and Tax Reporting](#)

Manage Users

Use these REST APIs to manage users.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-1 Manage Users

Task	Request	REST Resource
Add Users to an Identity Domain	POST	/interop/rest/ security/<api_version>/ users
Remove Users from an Identity Domain	DELETE	/interop/rest/ security/<api_version>/ users? filename=<filename>
Assign Users to Predefined or Application Roles	PUT	/interop/rest/ security/<api_version>/ users
Remove Users' Role Assignment	PUT	/interop/rest/ security/<api_version>/ users
Add Users To a Group	PUT	/interop/rest/security/ <api_version>/groups
Remove Users from a Group	PUT	/interop/rest/ security/<api_version>/ groups
Add a User To a Batch of Groups	PUT	/interop/rest/ security/<api_version>/ groups
Remove a User from a Batch of Groups	PUT	/interop/rest/ security/<api_version>/ groups
User Access Report (v1)	POST	/interop/rest/ {api_version}/reports? q={type:provisionreport ,fileName:provreport.cs v,format=simplified,use rtype=serviceusers}
User Access Report (v2)	POST	/interop/rest/v2/ reports/useraccess
User Group Report	POST	/interop/rest/ security/ <api_version>/ usergroupreport
User Audit Report (v1)	POST	/interop/rest/ {api_version}/reports? q={type:userauditrepo ,fileName:userauditrepo rt.csv,since=2017-12-10 ,until=2018-06-10}
User Audit Report (v2)	POST	/interop/rest/v2/ reports/useraudit

Table 10-1 (Cont.) Manage Users

Task	Request	REST Resource
User Role Assignment Report	POST	/interop/rest/security/ {api_version}/ roleassignmentreport/
Role Assignment Audit Report for OCI (Gen 2) Environments	PUT	/interop/rest/security/ {api_version}/ roleassignmentauditreport/
Invalid Login Report for OCI (Gen 2) Environments	PUT	/interop/rest/security/ {api_version}/ invalidloginreport/

Add Users to an Identity Domain

Creates a batch of users in an identity domain using an ANSI or UTF-8 encoded Comma Separated Value (CSV) file that was previously uploaded to the environment. The CSV file should not include the account of the user who executes this command. You can use the [Upload REST API](#) to upload the file. The file should be deleted after the API executes. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed. The file format is as follows:

```
First Name,Last Name,Email,User Login
Jane,Doe,jane.doe@example.com,jdoe
John,Doe, john.doe@example.com, john.doe@example.com
```

This API sends each new user an email with details about their accounts (user name and password) if `resetpassword` is set to `true`. If `resetpassword` is set to `false`, the email is not sent. If you set `resetpassword` to `false`, you should specify `userpassword`. Otherwise, a unique temporary password will be assigned to each user, but, because no email is sent, the passwords will not be known to the users so they will not be able to log in.

See [Importing a Batch of User Accounts](#) in *Getting Started with Oracle Cloud* for a detailed description of the CSV file format.

If a user definition in the CSV file matches a user account that exists in the identity domain, no changes will be made to the existing user account. This API creates accounts only for new users whose account information is included in the file. Because user accounts are common to all service environments that an identity domain supports, new users are available to all the environments that share the identity domain.

This API should be run only by an Identity Domain Administrator in the identity domain where users are to be created. In addition, the user running the API must also be the Service Administrator of the environment where the API is targeted to.

The API is asynchronous and returns the Job ID. The presence of status `-1` in the response indicates that the creation of users is in progress. Use the job status URI to determine whether the creation of users is complete. Any non-zero status except `-1` indicates failure of adding users.

This REST API is version v1.

Required Roles

Service Administrator and Identity Domain Administrator

Table 10-2 Tasks for Add Users

Task	Request	REST Resource
Add users	POST	/interop/rest/security/<api_version>/users
Add users status	GET	/interop/rest/security/<api_version>/jobs/<jobid>

REST Resource

POST /interop/rest/security/<api_version>/users

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the POST request parameters.

Table 10-3 Parameters

Name	Description	Type	Required	Default
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing the users to add, such as addUsers.csv.	Form	Yes	None
userpassword	Optionally, indicates the default password that you want to assign to all the new users who are created in the identify domain. If specified, this password must meet the minimum identity domain password requirements. If specified, the value of the userpassword parameter is used as the password for all users specified in the CSV file. Assigning the same password to all users may be desirable if you are creating users purely for testing purposes. If you are creating real EPM Cloud users and want to assign a specific password to each user, use this command without specifying a valid for the userpassword optional parameter.	Form	No	None

Table 10-3 (Cont.) Parameters

Name	Description	Type	Required	Default
resetpassword	<p>Optionally, indicates whether new users must change password at the first login. Unless this parameter is set to <code>false</code>, new users will be forced to change the password at the first login.</p> <p>This parameter sends each new user an email with details about their accounts (user name and password) if <code>resetPassword</code> is set to <code>true</code>.</p> <p>If <code>resetPassword</code> is set to <code>false</code>, the email is not sent.</p> <p>Note: If you set <code>resetPassword</code> to <code>false</code>, you should specify <code>userPassword</code>. Otherwise, a unique temporary password will be assigned to each user, but, because no email is sent, the passwords will not be known to the users, so they will not be able to log in.</p>	Form	No	<code>true</code>

Response

Supported Media Types: `application/json`

Parameters:

Table 10-4 Parameters

Name	Description
<code>details</code>	In the case of errors, details are published with the error string
<code>status</code>	See Migration Status Codes
<code>links</code>	Detailed information about the link
<code>href</code>	Links to API call or status API
<code>action</code>	The HTTP call type
<code>rel</code>	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status
<code>data</code>	Parameters as key value pairs passed in the request
<code>items</code>	Details about the resource
<code>links</code>	Details of the first URL to be requested to get the job details; <code>rel</code> is "Job Details"

Note: This API assigns one password (value of `userpassword`) to all the users specified in the CSV file. Assigning the same password to all users may be desirable if you are creating users purely for testing purposes.

If you are creating real EPM Cloud users and want to assign a specific password to each user, use this API for a single user at a time. That is, specify a single user in the CSV file and provide a password for this user in the API. Then, specify the other user in the CSV file and provide a different password for this user in the API.

When you add users using this API, unlike when you add users from My Services, Oracle Cloud does not send automatic emails to each newly added user. You should manually email credentials (login name and password) to each new user. Additionally, you should force new users to change password at first login by specifying `resetpassword` as `true`.

Examples of Response Body in JSON format.

Example 1, when job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/users",
      "data": {
        "jobType": "ADD_USERS",
        "filename": "<filename>",
        "resetpassword": "<true|false>"      },
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/users",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Example 2, when job completes with errors:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/",
      "data": null,
      "action": "GET"
    }
  ],
}
```

```

    "details": "Failed to add users. Input file <fileName> is not found.
Specify a valid file name.",
    "status": 1,
    "items": null
}

```

Example 3, when job completes without errors:

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "UserName": "<USERNAME>", "Error_Details": "User <USERNAME>
already exists. Please provide a different user name."
    }
  ]
}
)

```

Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void addUsers(String fileName, String userPassword, boolean
resetPassword) {
    try {
        String url = this.serverUrl + "/interop/rest/security/
<api_version>/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("userpassword", userPassword);
        reqParams.put("resetpassword", resetPassword + "");

        Map<String, String> resetResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =

```

```
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
    System.out.println(jobStatus);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```
funcAddUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&userpassword=$2&resetpassword=$3"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AddUsers"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}
```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def addUsers(fileName, resetPassword, userPassword) {

    String scenario = "Creating users in " + fileName;
    String params = "jobtype=ADD_USERS&filename="+ fileName
+"&resetpassword="+ resetPassword + "&userpassword="+ userPassword;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}
```

Remove Users from an Identity Domain

Deletes the identity domain accounts identified in an ANSI or UTF-8 encoded CSV file that was uploaded to the environment. Before running this command, use the [Upload REST API](#) to upload the file. The file format is as follows:

```
User Login
jane.doe@example.com
jdoe@example.com
```

This API should be run only by Service Administrators who are also assigned to the Identity Domain Administrator role in the identity domain from which users are to be removed. The CSV file should not include the account of the user who executes this command. Because user accounts are common to all service environments that an Identity Domain Administrator supports, deleting an account for one environment deletes it for all environments that share the Identity Domain Administrator. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

The API is asynchronous and returns the Job ID. The presence of status -1 in the response indicates that the removal of users is in progress. Use the job status URI to determine whether the removal of users is complete. Any non-zero status except -1 indicates failure of removing users.

Required Roles

Service Administrator and Identity Domain Administrator

This API is version v1.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-5 Tasks for Remove Users

Task	Request	REST Resource
Remove users	DELETE	/interop/rest/security/<api_version>/users?filename=<filename>
Remove users status	GET	/interop/rest/security/<api_version>/jobs/

REST Resource

DELETE /interop/rest/security/<api_version>/users?filename=<filename>

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the DELETE request parameters.

Table 10-6 Parameters

Name	Description	Type	Required	Default
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file name of a CSV file containing the login names of the users to be removed, for example, removeUsers.csv.	Query	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 10-7 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body in JSON format

Example 1: Response when the job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/users?filename=<filename>",
      "data": {
        "jobType": "REMOVE_USERS",
        "filename": "<filename>"
      },
      "action": "DELETE"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
```

```
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/  
<api_version>/jobs/",  
  "data": null,  
  "action": "GET"  
}  
],  
"details": null,  
"status": -1,  
"items": null  
}
```

Example 2: Response when the job completes with errors

```
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/  
<api_version>/jobs/",  
      "data": null,  
      "action": "GET"  
    }  
  ],  
  "details": "Failed to remove users. Input file <filename> is not found.  
Specify a valid file name.",  
  "status": 1,  
  "items": null  
}
```

Example 3: Response when the job completes with no errors

```
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>..oraclecloud.com/interop/rest/security/  
<api_version>/jobs/",  
      "data": null,  
      "action": "GET"  
    }  
  ],  
  "details": "Processed - 3, Succeeded - 1, Failed - 2.",  
  "status": 0,  
  "items": [  
    {  
      "UserName": "<USERNAME>", "Error_Details":  
"User <USERNAME> is not found. Verify that the user exists."  
    }  
  ],  
}
```


Example 10-1 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#).

```

public void removeUsers(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/
<api_version>/users";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "DELETE");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-2 Shell Script Sample CodePrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [CSS Common Helper Functions for cURL](#).

```

funcRemoveUsers() {
    url="$SERVER_URL/interop/rest/security/<api_version>/users"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="RemoveUsers"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code[CSS Common Helper Functions for Groovy](#).

```

def deleteUsers(fileName) {

```

```

String scenario = "Deleting users in " + fileName;
String params = null;
def url = null;
def response = null;
try {
    url = new URL(serverUrl + "/interop/rest/security/<api_version>/
users?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Assign Users to a Predefined Role or Application Role

This API assigns users included in an ANSI or UTF-8 encoded CSV file to a pre-defined or application role. Use this API to assign users (including the user who invokes this API) to a pre-defined role or to assign a user with application roles.

To assign a user to an application role, that user should already have a pre-defined role assigned to them.

Use double quotation marks to enclose role names that contain space characters in the CSV file. Before using this API, use the [Upload REST API](#) to upload files to the environment. The file should be deleted after the API executes.

The file format is as follows:

```

User Login
jane.doe@example.com
jdoe

```

The API is asynchronous and returns the Job ID. The presence of status -1 in the response indicates that assigning users is in progress. Use the job status URI to determine whether the assignment of roles is complete. Any non-zero status except -1 indicates failure of assigning users. With this API, you can see which records failed and the reason why they failed, in addition to how many records passed and failed.

This API is version v1.

Required Roles

Service Administrator

Table 10-8 Tasks for Assign Users to Roles

Task	Request	REST Resource
Assign role	PUT	/interop/rest/security/<api_version>/users
Assign role status	GET	/interop/rest/security/<api_version>/jobs/<jobid>

REST Resource

PUT /interop/rest/security/<api_version>/users

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the PUT request parameters.

Table 10-9 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	ASSIGN_ROLE	Form	Yes	None
filename	The name of the ANSI or UTF-8 encoded CSV file containing the login IDs of the users whose role assignment is to be modified, such as assignRoles.csv.	Form	Yes	None

Table 10-9 (Cont.) Parameters

Name	Description	Type	Required	Default
rolename	<p>The name of a pre-defined or application role applicable to the service. An incorrect role name will result in an error.</p> <p>It identifies one of the following:</p> <ul style="list-style-type: none"> • If you are assigning users to a pre-defined identity domain role, roleName should identify a pre-defined role applicable to the service. See <i>Understanding Predefined Roles in Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>. • Acceptable values for services other than Oracle Enterprise Data Management Cloud: <ul style="list-style-type: none"> – Service Administrator – Power User – User (do not use Planner, which was used in earlier versions of the service) – Viewer • Acceptable values for Oracle Enterprise Data Management Cloud: <ul style="list-style-type: none"> – Service Administrator – User • If you are assigning users to an application role, roleName should identify an application role listed in the assign roles tab of Access Control. Acceptable values for FreeForm, Planning, Planning Modules, Sales Planning, Strategic Workforce Planning, Financial Consolidation and Close, and Tax Reporting applications: <ul style="list-style-type: none"> – Approvals Administrator – Approvals Ownership Assigner – Approvals Supervisor – Approvals Proess Designer – Ad Hoc Grid Creator – Ad Hoc User – Ad Hoc Read Only User – Calculation Manager Administrator – Create Integration 	Form	Yes	None

Table 10-9 (Cont.) Parameters

Name	Description	Type	Required	Default
	<ul style="list-style-type: none"> - Drill Through - Run Integration - Mass Allocation - Task List Access Manager 			
	<p>Acceptable values for Account Reconciliation :</p> <ul style="list-style-type: none"> - Manage Alert Types - Manage Announcements - Manage Data Loads - Manage Organizations - Manage Periods - Manage Profiles and Reconciliations - Reconciliation Manage Currencies - Reconciliation Manage Public Filters and Lists - Reconciliation Manage Reports - Reconciliation Manage Teams - Reconciliation Manage Users - Reconciliation Commentator - Reconciliation Preparer - Reconciliation Reviewer - Reconciliation View Jobs - Reconciliation View Profiles - View Audit - View Periods 			
	<ul style="list-style-type: none"> • Acceptable values for Oracle Enterprise Data Management Cloud applications: <ul style="list-style-type: none"> - Application Creator - Auditor - View Creator • Acceptable values for Oracle Enterprise Profitability and Cost Management applications: <ul style="list-style-type: none"> - Ad Hoc Grid Creator - Ad Hoc Read Only User - Ad Hoc User - Clear POV Data - Copy POV Data - Create/Edit Rule - Create Integration - Create Model - Create POV 			

Table 10-9 (Cont.) Parameters

Name	Description	Type	Required	Default
	<ul style="list-style-type: none"> - Create Profit Curve - Delete Calculation History - Delete Model - Delete POV - Delete Rule - Drill Through - Edit POV Status - Edit Profit Curve - Mass Edit of Rules - Run Calculation - Run Integration - Run Profit Curve - Run Rule Balancing - Run Trace Allocation - Run Validation - View Calculation History - View Model 			
	<ul style="list-style-type: none"> • Acceptable values for Oracle Enterprise Profitability and Cost Management applications: <ul style="list-style-type: none"> - Ad Hoc Grid Creator - Ad Hoc Read Only User - Ad Hoc User - Clear POV Data - Copy POV Data - Create/Edit Rule - Create Integration - Create Model - Create POV - Create Profit Curve - Delete Calculation History - Delete Model - Delete POV - Delete Rule - Drill Through - Edit POV Status - Edit Profit Curve - Mass Edit of Rules - Run Calculation - Run Integration - Run Profit Curve - Run Rule Balancing - Run Trace Allocation - Run Validation - View Calculation History - View Model 			
	<p>For a description of these roles, see <i>Managing Role Assignments at the Application Level</i> in</p>			

Table 10-9 (Cont.) Parameters

Name	Description	Type	Required	Default
	<i>Administering Access Control for Oracle Enterprise Performance Management Cloud.</i>			

Response

Supported Media Types: application/json

Parameters:

Table 10-10 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body in JSON format

Example 1, when the job is in progress:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/users",
      "data": {
        "jobType": "ASSIGN_ROLE",
        "filename": "<filename>",
        "rolename": "<rolename>"
      },
      "action": "PUT"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
```

```
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/  
<api_version>/jobs/<jobid>",  
  "data": null,  
  "action": "GET"  
}  
],  
"details": null,  
"status": -1,  
"items": null  
}
```

Example 2, when the job completes with errors

```
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/  
<api_version>/jobs/<jobid>",  
      "data": null,  
      "action": "GET"  
    }  
  ],  
  "details": " Failed to assign role for users. Input file <filename> is not  
found. Specify a valid file name.",  
  "status": 1,  
  "items": null  
}
```

Example 3, when the job completes without errors

```
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/  
<api_version>/jobs/<jobid>",  
      "data": null,  
      "action": "GET"  
    }  
  ],  
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",  
  "status": 0,  
  "items": [  
    {  
      "UserName": "<USERNAME>", "Error_Details": "User <USERNAME> is  
not found. Verify that the user exists."  
    }  
  ]  
}
```


Example 10-3 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#).

```

public void assignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ASSIGN_ROLE");
        reqParams.put("rolename", roleName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-4 Shell Script Sample CodePrerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)Common Functions: See [CSS Common Helper Functions for cURL](#).

```

funcAssignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=ASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="AssignRole"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#).

```

def assignUsersRoles(fileName, roleName) {

    String scenario = "Assigning users in " + fileName + " with role " +
roleName;
    String params = "jobtype=ASSIGN_ROLE&filename="+ fileName +"&rolename="+
roleName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

```

Remove Users' Role Assignment

Removes one role currently assigned to the users (including the user who invokes this API) whose login IDs are included in the ANSI or UTF-8 encoded CSV file that is used with this command. Before running this API, upload the file to the environment using the [Upload REST API](#). The file should be deleted after the API executes. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

Use double quotation marks to enclose role names that contain the space character.

The API is asynchronous and returns the Job ID. The presence of status -1 in the response indicates that the removal of role assignments is in progress. Use the job status URI to determine whether unassigning roles is complete. Any non-zero status except -1 indicates failure of unassigning roles.

This REST API is version v1.

Required Roles

Service Administrator

Table 10-11 Tasks for Unassign Users to Roles

Task	Request	REST Resource
Unassign role	PUT	/interop/rest/security/<api_version>/users

Table 10-11 (Cont.) Tasks for Unassign Users to Roles

Task	Request	REST Resource
Unassign role status	GET	/interop/rest/security/<api_version>/jobs/<jobid>

REST Resource

PUT /interop/rest/security/<api_version>/users

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the PUT request parameters.

Table 10-12 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	UNASSIGN_ROLE	Form	Yes	None
filename	<p>The name of the ANSI or UTF-8 encoded CSV file containing the users whose role assignment is to be revoked, such as <code>unsignRole.csv</code>.</p> <p>The CSV file must have been uploaded already using the Upload REST API. The CSV file should not include the account of the user who executes this command.</p> <p>File format example:</p> <pre>User Login <email> <FirstName2.LastName2></pre>	Form	Yes	None

Table 10-12 (Cont.) Parameters

Name	Description	Type	Required	Default
rolename	<p>The name of a pre-defined or application role applicable to the service. An incorrect role name will result in an error.</p> <p>It identifies one of the following:</p> <ul style="list-style-type: none"> • If you are removing users from a pre-defined role, roleName should identify a pre-defined role applicable to the service. See <i>Understanding Predefined Roles in Getting Started with Oracle Enterprise Performance Management Cloud for Administrators</i>. • Acceptable values for services other than Oracle Enterprise Data Management Cloud: <ul style="list-style-type: none"> – Service Administrator – Power User – User (do not use Planner, which was used in earlier versions of the service) – Viewer • Acceptable values for Oracle Enterprise Data Management Cloud: <ul style="list-style-type: none"> – Service Administrator – User • If you are removing users from an application role, roleName should identify an application role listed in the assign roles tab of Access Control. Acceptable values for Oracle Planning, Oracle Planning Modules, Oracle Financial Consolidation and Close, Sales Planning, Strategic Workforce Planning, and Oracle Tax Reporting applications: <ul style="list-style-type: none"> – Approvals Administrator – Approvals Ownership Assigner – Approvals Process Designer – Approvals Supervisor – Ad Hoc Grid Creator – Ad Hoc User – Ad Hoc Read Only User 	Form	Yes	None

Table 10-12 (Cont.) Parameters

Name	Description	Type	Required	Default
	<ul style="list-style-type: none"> – Calculation Manager Administrator – Create Integration – Drill Through – Run Integration – Mass Allocation – Task List Access Manager • Acceptable values for Account Reconciliation applications: <ul style="list-style-type: none"> – Manage Alert Types – Manage Announcements – Manage Data Loads – Manage Organizations – Manage Periods – Manage Profiles and Reconciliations – Reconciliation Manage Currencies – Reconciliation Manage Public Filters and Lists – Reconciliation Manage Reports – Reconciliation Manage Teams – Reconciliation Manage Users – Reconciliation Commentator – Reconciliation Preparer – Reconciliation Reviewer – Reconciliation View Jobs – Reconciliation View Profiles – View Audit – View Periods • Acceptable values for Oracle Enterprise Data Management Cloud applications: <ul style="list-style-type: none"> – Application Creator – Auditor – View Creator • Acceptable values for Oracle Enterprise Profitability and Cost Management applications: <ul style="list-style-type: none"> – Ad Hoc Grid Creator – Ad Hoc Read Only User 			

Table 10-12 (Cont.) Parameters

Name	Description	Type	Required	Default
	<ul style="list-style-type: none"> – Ad Hoc User – Clear POV Data – Copy POV Data – Create/Edit Rule – Create Integration – Create Model – Create POV – Create Profit Curve – Delete Calculation History – Delete Model – Delete POV – Delete Rule – Drill Through – Edit POV Status – Edit Profit Curve – Mass Edit of Rules – Run Calculation – Run Integration – Run Profit Curve – Run Rule Balancing – Run Trace Allocation – Run Validation – View Calculation History – View Model 			
	<p>For a description of these roles, see <i>Managing Role Assignments at the Application Level</i> in Administering Access Control for Oracle Enterprise Performance Management Cloud .</p>			

Response

Supported Media Types: application/json

Parameters:

Table 10-13 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type

Table 10-13 (Cont.) Parameters

Name	Description
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is Job Details

Examples of the Response Body in JSON format

Example 1, when the job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/users",
      "data": {
        "jobtype": "UNASSIGN_ROLE",
        "filename": "<fileName>",
        "rolename": "<roleName>"
      },
      "action": "PUT"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobid>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Example 2, when the job completes with errors

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/
security/<api_version>/jobs/<jobid>",
      "data": null,

```

```

        "action": "GET"
    }
],
"details": "Failed to unassign role for users. Input file <filename> is
not found. Specify a valid file name.",
"status": 1,
"items": null
}

```

Example 3, when the job completes without errors

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/security/
<api_version>/jobs/<jobid>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "UserName": "<USERNAME>", "Error_Details": "User <USERNAME> is
not found. Verify that the user exists."
    }
  ]
}

```

Example 10-5 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void unassignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "UNASSIGN_ROLE");
        reqParams.put("rolename", roleName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,

```



```

        "PUT");
        String jobStatus =
        CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-6 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#).

```

funcUnassignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=UNASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="UnassignRole"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#).

```

def unassignUsersRoles(fileName, roleName) {

    String scenario = "Un-assigning users in " + fileName + " with
role " + roleName;
    String params = "jobtype=UNASSIGN_ROLE&filename="+ fileName
+"&rolename="+ roleName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

```

Update Users

Updates a batch of existing users in an identity domain using an ANSI or UTF-8 encoded comma-separated value (CSV) file that was uploaded to the environment. Before using this API, use the [Upload REST API](#) to upload the file. Use double quotation marks to enclose fields that contain space characters in the CSV file. The file should be deleted after the API executes. The file format is as follows:

```
Firt Name, Last Name,Email, User Login
Jane,Doe,<emailAddress>,jdoe
John,Doe,<emailAddress>,<emailAddress>
```

This API should be run only by Service Administrators who are also assigned to the Identity Domain Administrator role in the identity domain in which users are to be updated. The CSV file should not include the account of the user who executes this command. It updates all properties of the user identified by *User Login*. Because user accounts are common to all service environments that an Identity Domain Administrator supports, updating an account for one environment updates it for all environments that share the Identity Domain.

With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

The API is asynchronous and returns the Job ID. The presence of status -1 in the response indicates that the updating of users is in progress. Use the job status URI to determine whether the process is complete. Any non-zero status except -1 indicates failure.

This REST API is version v1.

Required Roles

Service Administrator and Identity Domain Administrator

Table 10-14 Tasks for Updating Users

Task	Request	REST Resource
Update users	PUT	/interop/rest/security/<api_verion>/users
Update users status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

PUT /interop/rest/security/<api_verion>/users

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-15 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	UPDATE_USERS	Form	Yes	None

Table 10-15 (Cont.) Parameters

Name	Description	Type	Required	Default
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing the users to update, such as updateUsers.csv.	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 10-16 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Examples of Response Body in JSON format.

Example 1, when job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/
security/<api_version>/users",
      "data": {
        "jobType": "UPDATE_USERS",
        "filename": "<filename>"
      },
      "action": "UPDATE"
    },
    {
      "rel": "Job Status",
      "href": " https://<SERVICE_NAME>-
```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/security/
<api_version>/jobs/<jobID>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}

```

Example 2, when job completes with errors:

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Failed to update users. Input file <filename> not found. Specify
a valid file name.",
  "status": 1,
  "items": null
}

```

Example 3, when job completes without errors:

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "UserName": "<username>",
      "Error_Details": " User <USER_NAME> not found. Verify that the
user exists. "
    }
  ]
}

```

```
    }
  }}

```

Example 10-7 Java Sample Code

Prerequisites: json.jar

Common Functions: See: [CSS Common Helper Functions for Java](#)

```

public void updateUsers(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
        apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "UPDATE_USERS");

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-8 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcUpdateUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=UPDATE_USERS"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="UpdateUsers"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Example 10-9 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def updateUser(fileName) {  
  
    String scenario = "Updating users from " + fileName ;  
    String params = "jobtype=UPDATE_USERS&filename="+ fileName;  
    def url = null;  
    def response = null;  
    try {  
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"  
users");  
    } catch (MalformedURLException e) {  
        println "Please enter a valid URL"  
        System.exit(0);  
    }  
    response = executeRequest(url, "PUT", params, "application/x-www-form-  
urlencoded");  
    if (response != null) {  
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),  
"GET");  
    }  
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Add Users To a Group

Adds a batch of users to an existing group in Access Control using an ANSI or UTF-8 encoded CSV file that was uploaded to the environment. Use the [Upload REST API](#) to upload the file. The file should be deleted after the API executes. The file format is as follows:

```
User Login  
<user name>  
<email>
```

Note: The user is added to the group only if both these conditions are met:

- User login IDs included in the file exist in the identity domain that services the environment
- The user is assigned to a pre-defined role in the identity domain

This API should be run only by a service administrator in the identity domain where users are to be added to the group.

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the assignment of users to the group is complete. The presence of status -1 in the response indicates that the addition of users to a group is in progress. Any non-zero status

except -1 indicates failure of adding users. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

Required Roles

Service Administrator

Table 10-17 Tasks for Add Users to Group

Task	Request	REST Resource
Add users to group	PUT	/interop/rest/security/<api_version>/groups
Add users to group status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

PUT /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-18 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	The string should have the value ADD_USERS_TO_GROUP. This value denotes that the users are being added to the group.	Form	Yes	None
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing the users to add, such as addUsersToGroup.csv. The file must have been uploaded already using the Upload REST API .	Form	Yes	None
groupname	The name of group to which the users must be added. This group must be a pre-existing group.	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 10-19 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; <code>rel</code> is "Job Details"

Examples of Response Body in JSON format.

Example 1, when job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/groups",
      "data": {
        "jobType": "ADD_USERS_TO_GROUP",
        "filename": "<fileName>",
        "groupName": "<groupName>",
      },
      "action": "GET"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```


Example 2, when job completes with errors:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobID>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Failed to add users to group. Input file <fileName> is
not found. Specify a valid file name.",
  "status": 1,
  "items": null
}
```

Example 3, when job completes without errors:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "UserName": "<USERNAME>", "Error_Details": "User
<USERNAME> is not found. Verify that the user exists."
    }
  ]
}
```

Example 10-10 Java Sample Code

Prerequisites: json.jar

Common Functions: See: [CSS Common Helper Functions for Java](#)

```
public void addUsersToGroup(String fileName, String groupName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
```

```

        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ADD_USERS_TO_GROUP");
        reqParams.put("groupname", groupName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-11 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcAddUsersToGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USERS_TO_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
    cssRESTAPI="AddUsersToGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Example 8-15 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#).

```

def addUsersToGroup(fileName, groupName) {

    String scenario = "Adding users in " + fileName + " to group " +
groupName;
    String params = "jobtype=ADD_USERS_TO_GROUP&filename="+ fileName
+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
}

```

```
        if (response != null) {
            getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
        }
    }
}
```

Example 10-12 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def addUsersToGroup(fileName, groupName) {

    String scenario = "Adding users in " + fileName + " to group " +
groupName;
    String params = "jobtype=ADD_USERS_TO_GROUP&filename="+ fileName
+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Remove Users from a Group

Removes users from a group listed in an ANSI or UTF-8 encoded CSV file from a group maintained in Access Control. You can use the [Upload REST API](#) to upload the file to the environment. The file format is as follows:

```
User Login
jdoe
john.doe@example.com
```

Note: A user is removed from a group only if both of these conditions are met:

- User logins included in the file exist in the identity domain that services the environment

- The user is assigned to a pre-defined role in the identity domain

This API can be run only by a service administrator in the identity domain from which users are to be removed.

The presence of status -1 in the response indicates that the removal of users is in progress. Use the job status URI to determine whether the removal of users is complete. Any non-zero status except -1 indicates failure of removing users. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-20 Tasks for Remove Users from Group

Task	Request	REST Resource
Remove users from group	PUT	/interop/rest/security/<api_version>/groups
Remove users from group status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

PUT /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-21 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	The string should have the value REMOVE_USERS_FROM_GROUP. This value denotes that the users are being removed from the group.			
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the users to be removed, for example, removeUsersFromGroup.csv. The file must have been uploaded already using the Upload REST API .	Form	Yes	None
groupname	The name of group from which the users must be removed. This group must be a pre-existing group.			

Response

Supported Media Types: application/json

Parameters:

Table 10-22 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body in JSON format

Example 1: Response when the job is in progress

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/groups",
      "data": {
        "jobType": "REST_REMOVE_USERS_FROM_GROUP",
        "filename": "<filename>"
        "groupName": "<groupName>"
      },
      "action": "PUT"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobID>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Example 2: Response when the job completes with errors

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>..oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Failed to remove users. Input file <filename> is not found.
Specify a valid file name.",
  "status": 1,
  "items": null
}

```

Example 3: Response when the job completes with no errors

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>..oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "UserName": "<USERNAME>", "Error_Details": "User <USERNAME> is not found.
Verify that the user exists."
    }
  ]
}

```

Example 10-13 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void removeUsersFromGroup(String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/
security/" + apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,

```

```

String>());
                                reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
                                .printBase64Bin
ary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

                                Map<String, String> reqParams = new
HashMap<String, String>();
                                reqParams.put("filename", fileName);

reqParams.put("jobtype", "REMOVE_USERS_FROM_GROUP");
                                reqParams.put("groupname", groupName);

                                Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                                "PUT");

                                String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult);
                                System.out.println(jobStatus);
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }
}

```

Example 10-14 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#).

```

funcRemoveUsersFromGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"

params="filename=$1&jobtype=REMOVE_USERS_FROM_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="RemoveUsersFromGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def removeUsersFromGroup(fileName, groupName) {

    String scenario = "Removing users in " + fileName + " from group "
+ groupName;
    String params = "jobtype=REMOVE_USERS_FROM_GROUP&filename="+
fileName + "&groupname="+ groupName;
    def url = null;

```

```
def response = null;
try {
    url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Add a User To a Batch of Groups

Adds an existing user to a batch of groups in Access Control using an ANSI or UTF-8 encoded CSV file that was uploaded to the environment. Use the [Upload REST API](#) to upload the file. The file should be deleted after the API executes. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed. The file format is as follows:

```
Group Name
GroupA
GroupB
```

The user is added to the groups only if these conditions are met:

- The user must exist in the identity domain that services the environment
- The user must be assigned to a pre-defined role in the identity domain
- The groups provided must exist in Access Control and must not be pre-defined groups

Additionally, the user running this API must be authorized to perform this action. This API should be run only by a service administrator in the environment where the user is to be added to the groups.

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the assignment of a user to the groups is complete. The presence of status -1 in the response indicates that the addition of a user to groups is in progress. Any non-zero status except -1 indicates failure of adding a user.

Required Roles

Service Administrator

This REST API is version v1.

Table 10-23 Tasks for Adding a User to a Batch of Groups

Task	Request	REST Resource
Add a user to groups	PUT	/interop/rest/security/<api_version>/groups
Add a user to groups status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

PUT /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-24 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	The string should have the value ADD_USER_TO_GROUPS. This value denotes that the user is being added to the groups.	Form	Yes	None
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing the groups to add the user to, such as addUserToGroups.csv. The file must have been uploaded already using the Upload REST API . File format example: Group Name GroupA GroupB	Form	Yes	None
username	The name of the user to add to the provided list of groups. This user must already exist.	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 10-25 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; <code>rel</code> is "Job Details"

Examples of Response Body in JSON format.

Example 1, when job is in progress

```
{
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/groups,
      "rel": "self",
      "data": {
        "jobType": "ADD_USER_TO_GROUPS",
        "filename": "<filename>",
        "username": "<username>"
      },
      "action": "PUT"
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>,
      "rel": "Job Status",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Example 2, when job completes with errors:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/v1/jobs/<jobID>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Failed to add user to groups. Input file <fileName> is
not found. Specify a valid file name.",
  "status": 1,
  "items": null
}
```

Example 3, when job completes without errors:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 2, Failed - 1.",
  "status": 0,
  "items": [
    {
      "GroupName": "<GROUPNAME>", "Error_Details": "Group
<GROUPNAME> is not found. Verify that the group exists."
    }
  ]
}
```

Example 10-15 Java Sample Code

Prerequisites: json.jar

Common Functions: See: [CSS Common Helper Functions for Java](#)

```
public void addUserToGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
```

```

        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ADD_USER_TO_GROUPS");
        reqParams.put("username", userName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-16 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcAddUserToGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USER_TO_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-urlencoded"
    cssRESTAPI="AddUserToGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Example 10-17 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def addUserToGroups(fileName, userName) {

    String scenario = "Adding users in " + fileName + " to group " +
userName;
    String params = "jobtype=ADD_USER_TO_GROUPS&filename="+ fileName
+"&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
}

```

```
response = executeRequest(url, "PUT", params, "application/x-www-  
form-urlencoded");  
if (response != null) {  
    getJobStatus(getUrlFromResponse(scenario, response, "Job  
Status"), "GET");  
}  
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Remove a User from a Batch of Groups

Removes a user from a batch of groups listed in an ANSI or UTF-8 encoded CSV file maintained in Access Control. You can use the [Upload REST API](#) to upload the file to the environment. The file format is as follows:

```
Group Name  
GroupA  
GroupB
```

A user is removed from groups only if these conditions are met:

- The user must exist in the identity domain that services the environment
- The user must be assigned to a pre-defined role in the identity domain
- The groups provided must exist in Access Control and must not be pre-defined groups

Additionally, the user running this API must be authorized to perform this action. This API should be run only by a service administrator in the environment where a user is to be removed from the groups. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

The presence of status -1 in the response indicates that the removal is in progress. Use the job status URI to determine whether the removal is complete. Any non-zero status except -1 indicates failure.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-26 Tasks for Remove a User from a Batch of Groups

Task	Request	REST Resource
Remove a user from groups	PUT	/interop/rest/security/<api_version>/groups
Remove a user from groups status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

PUT /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-27 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
jobtype	The string should have the value REMOVE_USER_FROM_GROUPS. This value denotes that the user is being removed from the groups.	Form	Yes	None
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the groups from which the user is to be removed, for example, removeUserFromGroups.csv. The file must have been uploaded already using the Upload REST API . File format: Group Name GroupA GroupB	Form	Yes	None
username	The name of the user to remove from the provided list of groups. This user must already exist.	Form	Yes	None

Response

Supported Media Types: application/json

Parameters:

Table 10-28 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link

Table 10-28 (Cont.) Parameters

Name	Description
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: self or Job Status. If the value is set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Example of Response Body in JSON format

Example 1: Response when the job is in progress

```
{
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/groups,
      "rel": "self",
      "data": {
        "jobType": "REMOVE_USER_FROM_GROUPS",
        "filename": "<filename>",
        "username": "<username>"
      },
      "action": "PUT"
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>,
      "rel": "Job Status",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Example 2: Response when the job completes with errors

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>..oraclecloud.com/interop/rest/
```

```

security/<api_version>/jobs/<jobId>",
    "data": null,
    "action": "GET"
  }
],
"details": "Failed to remove user from groups. File <filename> is not
found. Specify a valid file name.",
"status": 1,
"items": null
}

```

Example 3: Response when the job completes with no errors

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>..oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "data": null,
      "action": "GET"
    }
  ],
  "details": "Processed - 3, Succeeded - 1, Failed - 2.",
  "status": 0,
  "items": [
    {
      "GroupName": "<GROUPNAME>", "Error_Details": "Group
<GROUPNAME> is not found. Verify that the group exists."
    },
    {
      "GroupName": "<GROUPNAME>", "Error_Details": "Group
<GROUPNAME> is not found. Verify that the group exists."
    }
  ]
}

```

Example 10-18 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void removeUserFromGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();

```



```

        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "REMOVE_USER_FROM_GROUPS");
        reqParams.put("username", userName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-19 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#).

```

funcRemoveUserFromGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"

    params="filename=$1&jobtype=REMOVE_USER_FROM_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-urlencoded"
    cssRESTAPI="RemoveUserFromGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def removeUserFromGroups(fileName, userName) {

    String scenario = "Removing users in " + fileName + " from group "
+ userName;
    String params = "jobtype=REMOVE_USER_FROM_GROUPS&filename="+
fileName+"&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job

```

```

Status"), "GET");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Add Groups

Adds groups in Access Control using an ANSI or UTF-8 encoded CSV file that was uploaded to the environment. Use the [Upload REST API](#) to upload the file. The file should be deleted after the API executes. The file format is as follows:

```

Group Name,Description
GroupA,GroupADescription
GroupB,GroupBDescription

```

The user running this API must be authorized to perform this action. This API should be run only by a service administrator in the environment where groups are to be added. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether adding groups is complete. The presence of status -1 in the response indicates that the addition is in progress. Any non-zero status except -1 indicates failure of adding a group.

This REST API is version v1.

Required Roles

Service Administrator

Table 10-29 Tasks for Adding a Batch of Groups

Task	Request	REST Resource
Create groups	POST	/interop/rest/security/<api_version>/groups
Create groups status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

POST /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-30 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing the groups to add, such as <code>addGroups.csv</code> . The file must have been uploaded already using the Upload REST API . File format example: Group Name,Description GroupA,GroupADescription GroupB,GroupBDescription	Form	Yes	None

Response

Supported Media Types: `application/json`

Parameters:

Table 10-31 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the href to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; rel is "Job Details"

Examples of Response Body in JSON format.

Example 1, when job is in progress

```
{
  "links": [
    {
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/groups",
```

```

        "rel": "self",
        "data": {
            "jobType": "ADD_GROUPS",
            "filename": "<filename>"
        },
        "action": "POST"
    },
    {
        "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId ",
        "rel": "Job Status",
        "data": null,
        "action": "GET"
    }
],
"status": -1,
"details": null,
"items": null
}

```

Example 2, when job completes with errors:

```

{
    "links": [
        {
            "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/groups",
            "rel": "self",
            "data": {
                "jobType": "ADD_GROUPS",
                "filename": ""
            },
            "action": "POST"
        }
    ],
    "status": 1,
    "details": "EPMCSS-20671: Failed to create groups. Invalid or
insufficient parameters specified. Provide all required parameters for the
REST API. ",
    "items": null
}

```

Example 3, when job completes without errors:

```

{
    "links": [
        {
            "data": null,
            "action": "GET",
            "href": " http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/security/
<api_version>/jobs/<jobId>",

```

```

        "rel": "self"
    }
],
"status": 0,
"details": "Processed - 4, Succeeded - 3, Failed - 1.  ",
"items": [
{
    "GroupName": "<GROUPNAME>", "Error_Details": "Failed to
create a group with the name <GROUPNAME>. This group already exists in
the system. Provide a different group name."
}
]
}

```

Example 10-20 Java Sample Code

Prerequisites: json.jar

Common Functions: See: [CSS Common Helper Functions for Java](#)

```

public void addGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
        apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-21 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcAddGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
}

```

```

        header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
        cssRESTAPI="addGroups"
        statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
        echo $statusMessage
    }

```

Example 10-22 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def addGroups(fileName) {

    String scenario = "Creating Groups in " + fileName;
    String params = "filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Remove Groups

Removes groups listed in an ANSI or UTF-8 encoded CSV file maintained in Access Control. You can use the [Upload REST API](#) to upload the file to the environment. The file format is as follows:

```

Group Name
GroupA
GroupB

```

The user running this API must be authorized to perform this action. This API should be run only by a service administrator in the environment where a group is to be removed.

The presence of status -1 in the response indicates that the removal is in progress. Use the job status URI to determine whether the removal is complete. Any non-zero status except -1 indicates failure. With this API, you can see which records failed and the reason why they failed in addition to how many records passed and failed.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-32 Tasks for Remove Groups

Task	Request	REST Resource
Remove groups	DELETE	/interop/rest/security/<api_version>/groups
Remove groups status	GET	/interop/rest/security/<api_version>/jobs/<jobId>

REST Resource

DELETE /interop/rest/security/<api_version>/groups

Supported Media Types: application/x-www-form-urlencoded

Parameters:

The following table summarizes the request parameters.

Table 10-33 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
filename	The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the groups to be removed, for example, <code>removeGroups.csv</code> . The file must have been uploaded already using the Upload REST API .	Query	Yes	None
	Group Name GroupA GroupB			

Response

Supported Media Types: application/json

Parameters:

Table 10-34 Parameters

Name	Description
details	In the case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Possible values: <code>self</code> or <code>Job Status</code> . If the value is set to <code>Job Status</code> , you can use the <code>href</code> to get the status
data	Parameters as key value pairs passed in the request
items	Details about the resource
links	Details of the first URL to be requested to get the job details; <code>rel</code> is "Job Details"

Example of Response Body in JSON format

Example 1: Response when the job is in progress

```
{
  "status": -1,
  "items": null,
  "links": [
    {
      "href": " http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/groups?filename=<filename>",
      "rel": "self",
      "data": {
        "jobType": "REMOVE_GROUPS",
        "filename": "<filename>"
      },
      "action": "DELETE"
    },
    {
      "href": " http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "rel": "Job Status",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null
}
```

Example 2: Response when the job completes with errors

```
{
  "links": [
```



```

    {
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/groups",
      "rel": "self",
      "data": {
        "jobType": "REMOVE_GROUPS",
        "filename": ""
      },
      "action": "DELETE"
    }
  ],
  "status": 1,
  "details": "EPMCSS-20673: Failed to delete groups. Invalid or
insufficient parameters specified. Provide all required parameters for
the REST API. ",
  "items": null
}

```

Example 3: Response when the job completes with no errors

```

{
  "links": [
    {
      "data": null,
      "action": "GET",
      "href": " http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com /interop/rest/
security/<api_version>/jobs/<jobId>",
      "rel": "self"
    }
  ],
  "status": 0,
  "details": "Processed - 3, Succeeded - 2, Failed - 1.  ",
  "items": [
    {
      "GroupName": "<GROUPNAME>", "Error_Details": "Group
<GROUPNAME> is not found. Verify that the group exists."
    }
  ]
}

```

Example 10-23 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void removeGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();

```

```

        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "DELETE");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-24 Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#).

```

FuncRemoveGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="removeGroups"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def removeGroups(fileName) {

    String scenario = "Deleting Groups in " + fileName;
    String params = null;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-form-
urlencoded");
    if (response != null) {

```

```

        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

User Group Report

Generates a User Group Report of users in the system and writes the report to the filename provided. This report lists the direct or indirect membership of users assigned to the group. It can be downloaded using the Download API.

The report indicates whether the user assignment to group is direct (as member of a group) or indirect (as member of a group that is a child of a nested group). The report identifies the user's login name, first name, last name, email address, assigned group, and type of assignment in the following format. It is identical to the CSV version of the report created from the User Group Report tab in Access Control.

For example, assume that user jdoe is a member of group Test1, which is a child of nested group Test2. In this scenario, the report will display the following information for jdoe:

```

User, First Name, Last Name, Email, Direct, Group
jdoe, John, Doe, jdoe@example.com, Yes, test1
jdoe, John, Doe, jdoe@example.com, No, test2

```

This is an asynchronous job and returns the Job ID.

Required Roles

Service Administrator

This API is version v1.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-35 Tasks for User Group Report

Task	Request	REST Resource
User Group Report	POST	/interop/rest/ security/ <api_version>/ usergroupreport

Table 10-35 (Cont.) Tasks for User Group Report

Task	Request	REST Resource
User Group Report Status	GET	/interop/rest/ security/ <api_version>/jobs/ <jobId>

REST Resource

POST /interop/rest/security/<api_version>/usergroupreport

Request

Supported Media Types: application/x-www-form-urlencoded

The following table summarizes the request parameters.

Table 10-36 Parameters

Name	Description	Type	Required	Default
api_version	The specific API version, v1	Path	Yes	None
filename	The name of the file where the report is to be populated, such as userGroupReport.csv.	Form	Yes	None

Response

Supported Media Types: application/json

Table 10-37 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "details": null,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/usergroupreport",
      "rel": "self",
      "data": {
        "jobType": "GENERATE_USER_GROUP_REPORT",
        "filename": "<filename>"
      },
      "action": "POST"
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>",
      "rel": "Job Status",
      "data": null,
      "action": "GET"
    }
  ],
  "status": -1,
  "items": null
}
```

Response example 2 when the job completes with errors:

```
{
  "details": "Failed to generate User Group Report. File <filename>
already exists. Please provide different file name. ",
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>",
      "rel": "self",
      "data": null,
      "action": "GET"
    }
  ],
  "status": 1,
  "items": null
}
```

Response example 3 when the job completes without errors

```
{
  "details": null,
  "links": [
    {
```

```

        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobid>",
        "rel": "self",
        "data": null,
        "action": "GET"
    }
],
"status": 0,
"items": null
}

```

Sample Code

Example 10-25 Example Java Sample Code

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```

//
// BEGIN
//
public void generateUserGroupReport(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/usergroupreport";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
// END
//

```

Example 10-26 Example Shell Script Sample Code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
funcGenerateUserGroupReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
usergroupreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateUserGroupReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}
```

Example 10-27 Example Groovy Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def generateUserGroupReport(fileName) {
    String scenario = "Generating User Group Report in " + fileName;
    String params = "jobtype=GENERATE_USER_GROUP_REPORT&filename="+
fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/usergroupreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

User Access Report (v1)

Generates an access report of users in the system and writes the report to the filename provided. This report can then be downloaded using the download command.

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-38 User Access Report

Task	Request	REST Resource
User Access Report	POST	/interop/rest/{api_version}/reports?q={type:provisionreport,fileName:provreport.csv,format:simplified,usertype:serviceusers}

REST Resource

```
POST /interop/rest/{api_version}/reports?
q={type:provisionreport,fileName:provreport.csv,format:simplified,usertype,serviceusers}
```

Parameters:

The following table summarizes the request parameters.

Table 10-39 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
fileName	File where report is to be populated	Query	Yes	None
type	Type of report being generated: provisionreport	Query	Yes	None
format	The format of the csv file, classic or simplified	Query	No	classic
usertype	Whether to generate the report only for Identity Domain Administrators, IDAdmins or ServiceUsers	Query	No	ServiceUsers

Response

Supported Media Types: application/json

Table 10-40 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
{api_version}/reports?
q={type=provisionreport,fileName=provreport.csv,format=simplified,usert
ype=serviceusers}",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
reports/3180399797144693",
      "data": null,
      "action": "GET"
    }
  ],
  "status": -1,
  "details": null
}
```

Sample Code

Example 10-28 Java Sample – ProvisionReport.java

Prerequisites: json.jar

Common Functions: See [Common Helper Functions for Java](#)

```
//
// BEGIN
```

```
//
public void provisionReport (String fileName, String type) throws Exception {
    JSONObject params = new JSONObject();
    params.put("fileName", java.net.URLEncoder.encode(fileName));
    params.put("type", java.net.URLEncoder.encode(type));
    params.put("format", "simplified");
    params.put("usertype", "usertype", "serviceusers");

    String urlString = String.format("%s/interop/rest/%s/reports?q=%s",
serverUrl, lcmVersion, params.toString());
    String response = executeRequest(urlString, "POST", params.toString(),
"application/x-www-form-urlencoded");
    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"), "GET");
}
//
// END
//
```

Example 10-29 cURL Sample – provisionreport.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```
funcProvisionReport () {
    url=$SERVER_URL/interop/rest/$LCM_VERSION/reports/

    param=$(echo
"q={type:$reporttype,fileName:$fileName,format:$mode,usertype:$usertype}" |
sed -f urlencode.sed)

    url=$url?$param

    funcExecuteRequest "POST" $url $param "application/json"

    output='cat response.txt'
    status='echo $output | jq '.status''

    if [ $status == -1 ]; then
        echo "copying snapshot in progress"
        funcGetStatus "GET"
    else
        error='echo $output | jq '.details''
        echo "Error occured. " $error
    fi

    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
}
```

Example 10-30 Groovy Sample – provisionreport.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def provisionReport (fileName, type) {
    def url;

    JSONObject param = new
JSONObject();
    try {

        param.put("fileName", fileName);
        param.put("type", type);
        param.put("format", mode);
        param.put("usertype", usertype);

        url = new URL(serverUrl + "/interop/rest/" + lcmVersion +
"/reports?q=" + param.toString());
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", param.toString(),
"application/x-www-form-urlencoded");

    if (response != null) {
        getJobStatus(fetchPingUrlFromResponse(response, "Job
Status"), "GET");
    }
}
```

Additional Sample Code

[Java Sample](#)

[cURL Sample](#)

User Access Report (v2)

The User Access Report (v2) REST API generates an access report of users provisioned in the environment and writes the report to the filename provided. This report can then be downloaded using the download command. This is an asynchronous job and uses the job status URI to determine if the operation is complete.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-41 User Access Report

Task	Request	REST Resource
User Access Report	POST	/interop/rest/v2/reports/useraccess

REST Resource

POST /interop/rest/v2/reports/useraccess

Supported Media Types: application/json

Parameters:

The following table summarizes the request parameters.

Table 10-42 Parameters

Name	Description	Type	Required	Default
fileName	File where report is to be populated	Payload	Yes	None
format	The format of the csv file, classic or simplified	Payload	No	classic
usertype	Whether to generate the report only for Identity Domain Administrators, IDAdmins or ServiceUsers	Payload	No	ServiceUsers

Example URL and Payload

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/reports/useraccess

```
{
  "fileName": "provisionreport.csv",
  "parameters": {
    "format": "simplified",
    "usertype": "IDAdmins"
  }
}
```

Response

Supported Media Types: application/json

Table 10-43 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes

Table 10-43 (Cont.) Parameters

Parameters	Description
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	null

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
reports/useraccess",
      "action": "POST",
      "rel": "self",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/jobs/22747066997747363",
      "action": "GET",
      "rel": "Job Status",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt
-H 'Content-Type: application/json' -d
'{"fileName": "provisionreport.csv", "parameters":
{"format": "simplified", "usertype": "IDAdmins"}}
' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
reports/useraccess'
```

User Audit Report (v1)

Generates a user audit report in the system and writes the report to the filename provided. The output CSV file contains the first character as a Byte Order Mark(BOM) character `\uffeff`. The API writes an encrypted application identifier following the BOM character. This application identifier is written between double quotes. Headers for the CSV file follow the application identifier. The report contains the details regarding the users logged into the system in a given time range.

The generated CSV file is compressed and the output is a ZIP file. The file can be downloaded using the Download REST API.

This is an asynchronous command, so use the job status URI to determine whether the operation is complete.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-44 User Audit Report

Task	Request	REST Resource
User Audit Report	POST	<code>/interop/rest/{api_version}/reports?q={type:userauditreport,fileName:userauditreport.csv,since:2017-12-10,until:2018-06-10}</code>

REST Resource

POST `/interop/rest/{api_version}/reports?q={type:userauditreport,fileName:userauditreport.csv,since:2017-12-10,until:2018-06-10}`

Request

Supported Media Types: `application/x-www-form-urlencoded`

The following table summarizes the request parameters.

Table 10-45 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	Specific API version	Path	Yes	None
<code>fileName</code>	File where report is to be populated	Query	Yes	None
<code>since</code>	Report generation start date	Query	Yes	None

Table 10-45 (Cont.) Parameters

Name	Description	Type	Required	Default
until	Report generation end date	Query	Yes	None
type	Type of report being generated, provisionreport or userauditreport	Query	Yes	None

Response

Supported Media Types: application/json

Table 10-46 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.cominterop/rest/
{api_version}/reports?q={type:userauditreport,fileName:useraudit
report.csv,since:2017-12-10,until:2018-06-10}",
      "data": null,
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
reports/3180621025673301",
      "data": null,
      "action": "GET"
    }
  ],
  "status": -1,
}
```

```

    "details": null
}

```

User Audit Report Sample Code

Example 10-31 Java Sample – UserAuditReport.java

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

//
// BEGIN
//
public void userAuditReport (String fileName, String type, String since,
String until) throws Exception {
    JSONObject params = new JSONObject();
    params.put ("fileName", java.net.URLEncoder.encode(fileName));
    params.put ("type", java.net.URLEncoder.encode(type));
    params.put ("since", java.net.URLEncoder.encode(since));
    params.put ("until", java.net.URLEncoder.encode(until));

    String urlString = String.format ("%s/interop/rest/%s/reports?q=%s",
serverUrl, lcmVersion, params.toString());
    String response = executeRequest (urlString, "POST", params.toString(),
"application/x-www-form-urlencoded");
    waitForCompletion (fetchPingUrlFromResponse (response, "Job Status"));
//
// END
//

```

Example 10-32 cURL Sample – userauditreport.sh

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [Common Helper Functions for cURL](#)

```

funcUserAuditReport () {
    url=$SERVER_URL/interop/rest/$LCM_VERSION/reports/

    param=$(echo
"q={type:$reporttype,fileName:$fileName,since:$since,until:$until}" | sed -f
urlencode.sed)

    url=$url?$param

    funcExecuteRequest "POST" $url $param "application/json"

    output='cat response.txt'
    status='echo $output | jq '.status''
    if [ $status == -1 ]; then
        echo "copying snapshot in progress"
        funcGetStatus "GET"
    else
        error='echo $output | jq '.details''
    fi
}

```



```
        echo "Error occured. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Example 10-33 Groovy Sample – userauditreport.groovy

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def userAuditReport (fileName, type, since, until) {
    def url;
    JSONObject param = new JSONObject();
    try {

        param.put("fileName", fileName);
        param.put("type", type);
        param.put("since", since);
        param.put("until", until);

        url = new URL(serverUrl + "/interop/rest/" + lcmVersion +
"/reports?q=" + param.toString());
        } catch (MalformedURLException e) {
            println "Malformed URL. Please pass valid URL"
            System.exit(0);
        }
        response = executeRequest(url, "POST", param.toString(),
"application/x-www-form-urlencoded");

        if (response != null) {
            waitForCompletion(fetchPingUrlFromResponse(response, "Job
Status"));
        }
    }
}
```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

User Audit Report (v2)

The User Audit Report (v2) REST API generates a user audit report in the environment and writes the report to the filename provided. The output CSV file contains the first character as a Byte Order Mark(BOM) character `\u0000`. The API writes an encrypted application identifier following the BOM character. This application identifier is written between double quotes. Headers for the CSV file follow the application identifier. The report contains the details regarding the users logged into the environment in a given time range.

The generated CSV file is compressed and the output is a ZIP file. The file can be downloaded using the Download REST API.

This is an asynchronous command, so use the job status URI to determine whether the operation is complete.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-47 User Audit Report

Task	Request	REST Resource
User Audit Report	POST	/interop/rest/v2/reports/useraudit

REST Resource

POST /interop/rest/v2/reports/useraudit

Request

Supported Media Types: application/json

The following table summarizes the request parameters.

Table 10-48 Parameters

Name	Description	Type	Required	Default
fileName	File where report is to be populated	Payload	Yes	None
since	Report generation start date	Payload	Yes	None
until	Report generation end date	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/reports/useraudit
```

```
{
  "fileName": "userauditreport.csv",
  "since": "2022-10-01",
  "until": "2022-11-01"
}
```

Response

Supported Media Types: application/json

Table 10-49 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "details": null,
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
reports/useraudit",
      "action": "POST",
      "rel": "self",
      "data": null
    },
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
status/jobs/22747152577657842",
      "action": "GET",
      "rel": "Job Status",
      "data": null
    }
  ]
}
```

Sample cURL command

```
curl -X POST -s -u '<USERNAME>:<PASSWORD>' -o response.txt -D
respHeader.txt -H
'Content-Type:application/json' -d
'{"fileName":"userauditreport.csv","until":"2022-11-01",
"since":"2022-10-01"}' 'https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
reports/useraudit'
```

Role Assignment Report

Generates a Role Assignment Report (.CSV). This report lists the predefined roles (for example, Service Administrator) and application roles (for example, Approvals Ownership Assigner, Approvals Supervisor, Approvals Administrator, and Approvals Process Designer, which are Planning application roles) assigned to users. This report matches the CSV version of the Role Assignment Report generated from Access Control. The API writes the report to the filename provided, and the report can then be downloaded using the [Download REST API](#).

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

The presence of status -1 in the response indicates that the generation of Role Assignment Report is in progress. Use the job status URI to determine whether the generation of Role Assignment Report is complete. Any non-zero status except -1 indicates failure of generating Role Assignment Report.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-50 User Assignment Report

Task	Request	REST Resource
Role Assignment Report	POST	/interop/rest/security/{api_version}/roleassignmentreport/
Role Assignment Report Status	GET	/interop/rest/security/{api_version}/jobs/{jobId}

REST Resource

POST /interop/rest/security/{api_version}/roleassignmentreport

Parameters:

The following table summarizes the request parameters.

Table 10-51 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
filename	File name for the file where the report is to be populated, such as roleAssignmentReport.csv	Form	Yes	None

Response

Supported Media Types: application/json

Table 10-52 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following show examples of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "links": [
    {
      "data": {
        "jobType": "GENERATE_ROLE_ASSIGNMENT_REPORT",
        "filename": "<filename>"
      },
      "action": "POST",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobId>",
      "rel": "Job Status"
    }
  ],
  "status": -1,
  "details": null,
  "items": null
}
```

Response 2 example when job completes with errors:

```
{
  "links": [
    {
      "data": {
        "jobType": "GENERATE_ROLE_ASSIGNMENT_REPORT",
        "filename": " "
      },
      "action": "POST",
      "href": "https://<SERVICE_NAME>-
```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
{api_version}/roleassignmentreport",
    "rel": "self"
  }
],
"status": 1,
"details": "EPMCSS-20665: Failed to generate Role Assignment Report.
Invalid or insufficient parameters are specified. Provide all required
parameters for the REST API. ",
"items": null
}

```

Response 3 example when job completes without errors:

```

{
  "links": [
    {
      "data": null,
      "action": "GET",
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobID>",
      "rel": "self"
    }
  ],
  "status": 0,
  "details": null,
  "items": null
}

```

Example 10-34 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void generateRoleAssignmentReport(String filename) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentreport";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", filename);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    }
}

```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Example 10-35 Shell Script Sample code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```
funcGenerateRoleAssignmentReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}
```

Example 10-36 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def generateRoleAssignmentReport(filename) {

    String scenario = "Generating Role assignment report in " +
filename;
    String params =
"jobtype=GENERATE_ROLE_ASSIGNMENT_REPORT&filename="+ filename;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}
```

Role Assignment Audit Report for OCI (Gen 2) Environments

Users with a Service Administrator role can use this API to generate a Role Assignment Audit Report of users with their pre-defined and application roles on OCI (Gen 2) Environments. This allows you to automate reporting on users role and application role assignments. The report shows all the changes made to the predefined role and application role assignments within the provided time frame. This report can be generated for the previous 90 days from the current date. You can download the report using the Download REST API.

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

The presence of status -1 in the response indicates that the generation of the report is in progress. Use the job status URI to determine whether the generation of the report is complete. Any non-zero status except -1 indicates failure of generating the report.

This API is version v1.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-53 Role Assignment Audit Report for OCI (Gen 2) Environments

Task	Request	REST Resource
Role Assignment Audit Report	POST	/interop/rest/security/{api_version}/roleassignmentauditreport/
Role Assignment Audit Report Status	GET	/interop/rest/security/{api_version}/jobs/{jobId}

REST Resource

POST /interop/rest/security/{api_version}/roleassignmentauditreport

Parameters:

The following table summarizes the request parameters.

Table 10-54 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
from_date	The start date for the report (in YYYY-MM-DD format)	Form	Yes	None
to_date	The end date for the report (in YYYY-MM-DD format)	Form	Yes	None

Table 10-54 (Cont.) Parameters

Name	Description	Type	Required	Default
filename	CSV file where the report is to be populated, such as roleAssignmentAuditReport.csv	Form	Yes	None

Response

Supported Media Types: application/json

Sample Role Assignment Audit report

	A	B	C	D	E
1	User Name	Role	Action	Administrator	Date and Time
2	jane.doe@example.com	User	Assign	epm.admin@example.com	July 09, 2021 03:54:52 UTC
3	jane.doe@example.com	Run Integrations	Assign	epm.admin@example.com	July 09, 2021 03:54:52 UTC
4	john.doe@example.com	Service Administrator	Assign	epm.admin@example.com	July 09, 2021 03:51:28 UTC
5	john.smith@example.com	Power User	Unassign	epm.admin@example.com	July 09, 2021 03:53:04 UTC
6	john.smith@example.com	User	Assign	epm.admin@example.com	July 09, 2021 03:54:06 UTC

Information on deleted users who were previously assigned to predefined roles in the environment is listed with the display name (first and last name) of the user in the User Name column. In such cases, the Role column indicates the predefined role that the user had before the user's account was deleted. This change does not apply to application roles, if any, that were assigned to the deleted user; such assignments are shown with the User Login Name of the user. For an example, see the information in the red box in the following illustration.

....

	A	B	C	D	E
1	User Name	Role	Action	Administrator	Date and Time
2	Jane Doe	User	Assign	epm.admin@example.com	July 09, 2021 03:54:52 UTC
3	jane.doe@example.com	Run Integrations	Assign	epm.admin@example.com	July 09, 2021 03:54:52 UTC
4	john.doe@example.com	Service Administrator	Assign	epm.admin@example.com	July 09, 2021 03:51:28 UTC
5	john.smith@example.com	Power User	Unassign	epm.admin@example.com	July 09, 2021 03:53:04 UTC
6	john.smith@example.com	User	Assign	epm.admin@example.com	July 09, 2021 03:54:06 UTC

Table 10-55 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following show examples of the response body in JSON format.

Response 1 example when job is in progress:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/roleassignmentauditreport",
      "data": {
        "jobType": "GENERATE_ROLE_ASSIGNMENT_AUDIT_REPORT",
        "to_date": "<toDate>",
        "filename": "<filename>",
        "from_date": "<fromDate>"
      },
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/3023387588778806",
      "data": null,
      "action": "GET"
    }
  ],
  "details": null,
  "status": -1,
  "items": null
}
```

Response 2 example when job completes with errors:

```
{
  "links": [
    {
      "data": {
        "jobType": "GENERATE_ROLE_ASSIGNMENT_AUDIT_REPORT",
        "from_date": " ",
        "to_date": " ",
        "filename": " "
      },
      "action": "POST",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
{api_version}/roleassignmentauditreport",
      "rel": "self"
    }
  ],
  "status": 1,
  "details": "EPMCSS-20678: Failed to generate Role Assignment Audit
Report. Invalid or insufficient parameters specified. Provide all required
parameters for the REST API. ",
}
```

```

    "items": null
  }

```

Response 3 example when job completes without errors:

```

{
  "links": [
    {
      "data": null,
      "action": "GET",
      "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobID>",
      "rel": "self"
    }
  ],
  "status": 0,
  "details": null,
  "items": null
}

```

Example 10-37 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void generateRoleAssignmentAuditReport(String fromDate, String
toDate,String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentauditreport";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("from_date", fromDate);
        reqParams.put("to_date", toDate);
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,

        "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Example 10-38 Shell Script Sample code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```
funcGenerateRoleAssignmentAuditReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentauditreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentAuditReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}
```

Example 10-39 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```
def generateRoleAssignmentAuditReport(from_date,to_date,fileName) {

    String scenario = "Generating Role assignment audit report in " +
fileName;
    String params =
"jobtype=GENERATE_ROLE_ASSIGNMENT_AUDIT_REPORT&from_date="+from_date+"&to_dat
e="+to_date+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
roleassignmentauditreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}
```

Invalid Login Report for OCI (Gen 2) Environments

Users who have both a Service Administrator role and an Identity Domain Administrator role can use this API to generate an Invalid Login Report on OCI (Gen 2) environments. This allows you to automate reporting on unsuccessful login attempts. This report shows unsuccessful login attempts for users within the provided time frame. This report can be generated for the previous 90 days from the current date. You can download the report using the Download REST API. This report shows all the unsuccessful login attempts to the

corresponding Identity Cloud Service. These may not all be to this particular EPM Cloud instance.

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

The presence of status -1 in the response indicates that the generation of the report is in progress. Use the job status URI to determine whether the generation of the report is complete. Any non-zero status except -1 indicates failure of generating the report.

This API is version v1.

Required Roles

Service Administrator and Identity Domain Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 10-56 Invalid Login Report for OCI (Gen 2) Environments

Task	Request	REST Resource
Invalid Login Report	POST	/interop/rest/security/{api_version}/invalidloginreport/
Invalid Login Report Status	GET	/interop/rest/security/{api_version}/jobs/{jobId}

REST Resource

POST /interop/rest/security/{api_version}/invalidloginreport

Parameters:

The following table summarizes the request parameters.

Table 10-57 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
from_date	The start date for the report (in YYYY-MM-DD format)	Form	Yes	None
to_date	The end date for the report (in YYYY-MM-DD format)	Form	Yes	None
filename	CSV file where the report is to be populated, such as InvalidLoginReport.csv	Form	Yes	None

Response

Supported Media Types: application/json

Example report:

	A	B	C
1	User Name	IP Address	Access Date and Time
2	john.doe@example.com	xxx.xx.xx.xx5	July 15, 2021 11:14:58 UTC
3	jane.doe@example.com	xxx.xx.xx.xx9	July 15, 2021 11:14:58 UTC
4	john.smith@example.com	xxx.xx.xx.xx3	July 15, 2021 11:14:57 UTC

Table 10-58 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following show examples of the response body in JSON format.

Response 1 example when job is in progress:

```
Copy{
  "links": [
    {
      "data": {
        "jobType": "GENERATE_INVALID_LOGIN_REPORT",
        "from_date": "<fromDate>",
        "to_date": "<toDate>",
        "filename": "<filename>"
      },
      "action": "POST",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/security/
<api_version>/jobs/<jobId>",
      "rel": "Job Status"
    }
  ],
  "status": -1,
  "details": null,
  "items": null
}
```

Response 2 example when job completes with errors:

```
Copy{
  "links": [
    {
```

```

        "data": {
            "jobType": "GENERATE_INVALID_LOGIN_REPORT",
            "from_date": " ",
            "to_date": " ",
            "filename": " "
        },
        "action": "POST",
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/{api_version}/invalidloginreport",
        "rel": "self"
    }
],
    "status": 1,
    "details": "EPMCSS-20679: Failed to generate Invalid Login Report.
Invalid or insufficient parameters specified. Provide all required
parameters for the REST API. ",
    "items": null
}

```

Response 3 example when job completes without errors:

```

{
    "links": [
        {
            "data": null,
            "action": "GET",
            "href": " https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
security/<api_version>/jobs/<jobID>",
            "rel": "self"
        }
    ],
    "status": 0,
    "details": null,
    "items": null
}

```

Example 10-40 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

public void generateInvalidLoginReport(String fromDate, String toDate,
String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/invalidloginreport";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +

```

```

this.password).getBytes(Charset.defaultCharset()));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("from_date", fromDate);
        reqParams.put("to_date", toDate);
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

Example 10-41 Shell Script Sample code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcGenerateInvalidLoginReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
invalidloginreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="generateInvalidLoginReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

Example 10-42 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def generateInvalidLoginReport(from_date,to_date,fileName) {

    String scenario = "Generating Invalid Login report in" + fileName;
    String params =
"jobtype=GENERATE_INVALID_LOGIN_REPORT&from_date="+from_date+"&to_date="+to_d
ate+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
invalidloginreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
}

```



```

    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

```

Group Assignment Audit Report

Generates a group assignment audit report. The report contains details on the users and groups that were added to or removed from Access Control groups in a given date range. This report is in CSV format. Each row of the report provides the user or group that was added or removed, the group to which the user or group was added or removed from, the Service Administrator who performed the action, and the date and time when the action was completed. The API writes the report to the filename provided, and the report can then be downloaded using the [Download REST API](#).

This is an asynchronous job and uses the job status URI to determine if the operation is complete.

The presence of status -1 in the response indicates that the generation of Role Assignment Report is in progress. Use the job status URI to determine whether the generation of Role Assignment Report is complete. Any non-zero status except -1 indicates failure of generating Role Assignment Report.

This API is version v2.

Required Roles

Service Administrator

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /interop/rest/{api_version}/reports/groupaudit

Supported Media Type: application/json

Table 10-59 Tasks for Group Assignment Audit Report

Task	Request	REST Resource
Group Assignment Audit Report	POST	/interop/rest/ {api_version}/reports/ groupaudit
Group Assignment Audit Report Status	GET	/interop/rest/ {api_version}/jobs/ {jobId}

Parameters:

The following table summarizes the request parameters.

Table 10-60 Parameters

Name	Description	Type	Required	Default
api_version	Specific API version	Path	Yes	None
filename	The CSV file where the report is to be populated, such as groupAssignmentAuditReport.csv	Payload	Yes	None
from_date	The start date for the report (in YYYY-MM-DD format)	Payload	Yes	None
to_date	The end date for the report (in YYYY-MM-DD format)	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/
interop/rest/v2/reports/groupaudit
{
  "fileName": "groupauditreport_test.csv",
  "from_date": "2022-03-26",
  "to_date": "2022-05-30"
}
```

Response

Supported Media Types: application/json

Table 10-61 Parameters

Parameters	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following show an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.cominterop/rest/
{api_version}/reports/groupaudit",
    "data": null,
    "action": "POST"
  },
  {
    "rel": "Job Status",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v2/
jobs/3180621025673301",
    "data": null,
    "action": "GET"
  }
],
"status": -1,
"details": null
}

```

Example 10-43 Java Sample Code

Prerequisites: json.jar

Common Functions: See [CSS Common Helper Functions for Java](#)

```

//
//BEGIN
//
public void groupAssignmentAuditReport(String fileName, String
from_date, String to_date)
    throws Exception {
    JSONObject params = new JSONObject();
    params.put("fileName", fileName);
    params.put("from_date", from_date);
    params.put("to_date", to_date);

    String urlString = String.format("%s/interop/rest/%s/reports/
groupaudit", serverUrl, apiVersion);
    String response = executeRequest(urlString, "POST",
params.toString(), "application/json");
    getJobStatus(fetchPingUrlFromResponse(response, "Job Status"),
"GET");
}
//
// END
//

```

Example 10-44 Shell Script Sample code

Prerequisites: jq (<http://stedolan.github.io/jq/download/linux64/jq>)

Common Functions: See [CSS Common Helper Functions for cURL](#)

```

funcgroupAssignmentAuditReport () {
    url=$SERVER_URL/interop/rest/v2/reports/groupaudit
    fileName="groupAssignmentAuditReport.csv"
}

```

```

    from_date="2022-03-01"
    to_date="2022-05-30"

param="{\"fileName\": \"${fileName}\", \"from_date\": \"${from_date}\", \"to_date\": \"${to_date}\"}"
    funcExecuteRequest "POST" $url "$param" "application/json"
    output=$(cat response.txt)
    status=$(echo $output | jq '.status')
    echo "Status :$status"
    if [ $status == -1 ]; then
        echo "group assignment audit report generation in progress"
        funcGetStatus "GET"
    else
        error='echo $output | jq '.details''
        echo "Error occured. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Example 10-45 Groovy Sample Code

Common Functions: See [CSS Common Helper Functions for Groovy](#)

```

def groupAssignmentAuditReport (fileName, from_date, to_date) {
    String scenario = "Group Assignment Audit Report";
    def url;
    def payload = new JsonBuilder()
    payload fileName:fileName,
            from_date:from_date,
            to_date:to_date
    url = new URL(serverUrl + "/interop/rest/v2/reports/groupaudit");
    params=payload.toString();
    response = executeRequest(url, "POST", params, "application/
json");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Adding Users to a Team for Account Reconciliation

Adds Oracle Enterprise Performance Management Cloud users listed in a UTF8 formatted CSV file to an existing team in Access Control for Account Reconciliation. The file must be uploaded to the environment before using this API, and the file should be deleted after the API executes. Use the [Upload REST API](#) to upload the file.

A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:

```
User Login, primary_user
jdoe, yes
jane.doe@example.com, no
```

Note: The users are added only if both these conditions are met:

- User login IDs included in the file exist in the identity domain that services the environment
- The user is assigned to a pre-defined role in the identity domain

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the update is in progress. Any non-zero status except -1 indicates failure for the update.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

REST Resource

POST /armARCS/rest/{version}/jobs

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/

Request

Supported Media Types: application/x-www-form-urlencoded

Example of Request Body

The following table summarizes the parameters of the JSON request.

Table 10-62 Parameters

Name	Data type	Description
version	String	The version of the API you are developing with. For the current release, the version is v1.
jobName	String	The name of the job, ADD_USERS_TO_TEAM.

Table 10-62 (Cont.) Parameters

Name	Data type	Description
fileName	String	<p>The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the users to be added, for example, addUsersToTeam.csv.</p> <p>The file must have been uploaded already using the Upload REST API. The CSV file should not include the account of the user who executes this command.</p> <p>A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:</p> <pre>User Login, primary_user jdoe, yes jane.doe@example.com,no</pre>
teamName	String	The name of an existing team in Access Control, such as Team1

Example of Request body

```
{
  "jobName":"ADD_USERS_TO_TEAM",
  "parameters":{
    "fileName":"users.csv",
    "teamName":"Team1"
  }
}
```

Response

Supported Media Types: application/json

Parameters

Table 10-63 Parameters

Name	Description
status	<p>-1 - In Progress</p> <p>0 - Success</p> <p>1 - Failure</p>
details	In case of errors, details are published with the error string
descriptiveStatus	The status of the job, such as Completed or Error.
items	Collection of notification categories
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Possible value: self

Table 10-63 (Cont.) Parameters

Name	Description
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/
100000000053010",
      "action": "GET"
    }
  ],
  "status": -1,
  "type": "ARCS",
  "link": null,
  "items": null,
  "error": null
}
```

Prerequisites: json.jar

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Adding Users to a Team for Financial Consolidation and Close and Tax Reporting

Adds Oracle Enterprise Performance Management Cloud users listed in a UTF8 formatted CSV file to an existing team in Access Control. The file must be uploaded to the environment before using this API, and the file should be deleted after the API executes. Use the [Upload REST API](#) to upload the file.

A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:

```
User Login, primary_user
jdoe, yes
jane.doe@example.com, no
```

Note: The users are added only if both these conditions are met:

- User login IDs included in the file exist in the identity domain that services the environment
- The user is assigned to a pre-defined role in the identity domain

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the update is in progress. Any non-zero status except -1 indicates failure for the update.

Note that this feature uses a Planning REST API to run a job. Details about Planning REST APIs are described here: [Planning REST APIs](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/FCCS/fcmjobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/x-www-form-urlencoded

The following table summarizes the parameters.

Table 10-64 Parameters

Name	Description	Type	Required	Default
api_version	The version of the API you are developing with; for the current release, it is v3	Path	Yes	None
application	The name of the application, for example, FCCS or TRCS	Path	Yes	None

Example of Request Body

The following table summarizes the parameters of the JSON request.

Table 10-65 Parameters

Name	Description
jobName	The name of the job, ADD_USERS_TO_TEAM
fileName	<p>The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the users to be added, for example, addUsersToTeam.csv.</p> <p>The file must have been uploaded already using the Upload REST API. The CSV file should not include the account of the user who executes this command.</p> <p>A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:</p> <pre>User Login, primary_user jdoe, yes jane.doe@example.com, no</pre>
teamName	The name of an existing team in Access Control, for example, Team1

Example of Request body

```
{
  "jobName": "ADD_USERS_TO_TEAM",
  "parameters": {
    "fileName": "users.csv",
    "teamName": "Team1"
  }
}
```

Response

Supported Media Types: application/json

Parameters

Table 10-66 Parameters

Name	Description
jobName	ADD_USERS_TO_TEAM
jobID	The ID of the job, such as 100000000114040
status	-1 - In Progress 0 - Success 1 - Failure
details	In case of errors, details are published with the error string
descriptiveStatus	The status of the job, such as Completed or Error
items	Collection of notification categories
links	Detailed information about the link
href	Links to the API call

Table 10-66 (Cont.) Parameters

Name	Description
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "jobName":"ADD_USERS_TO_TEAM",
  "jobId":100000000114040,
  "descriptiveStatus":",
  "detail":"In Progress",
  "status":-1,
  "items":null,
  "links":[
    {
      "rel":"self",
      "href":"https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/
HyperionPlanning/rest/v3/applications/FCCS/fcmjobs/100000000114040",
      "action":"GET"
    }
  ]
}
```

Prerequisites: json.jar

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Removing Users from a Team for Account Reconciliation

Removes Oracle Enterprise Performance Management Cloud users listed in a UTF8 formatted CSV file from an existing team in Access Control for Account Reconciliation. The file must be uploaded to the environment before using this API, and the file should be deleted after the API executes. Use the [Upload REST API](#) to upload the file.

A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:

```
User Login, primary_user
jdoe, yes
jane.doe@example.com,no
```

Note: The users are removed only if both these conditions are met:

- User login IDs included in the file exist in the identity domain that services the environment
- The user is assigned to a pre-defined role in the identity domain

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the update is in progress. Any non-zero status except -1 indicates failure for the update.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

POST /armARCS/rest/{version}/jobs

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/

Required Roles

Service Administrator

Request

Supported Media Types: application/x-www-form-urlencoded

Example of Request Body

The following table summarizes the parameters of the JSON request.

Table 10-67 Parameters

Name	Data type	Description
version	String	The version of the API you are developing with. For the current release, the version is v1.
jobName	String	The name of the job, REMOVE_USERS_FROM_TEAM.
fileName	String	The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the users to be removed, for example, removeUsersFromTeam.csv. The file must have been uploaded already using the Upload REST API . The CSV file should not include the account of the user who executes this command. A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows: User Login, primary_user jdoe, yes jane.doe@example.com,no

Table 10-67 (Cont.) Parameters

Name	Data type	Description
teamName	String	The name of an existing team in Access Control, such as Team1

Example of Request body

```
{
  "jobName": "REMOVE_USERS_FROM_TEAM",
  "parameters": {
    "fileName": "users.csv",
    "teamName": "Team1"
  }
}
```

Response

Supported Media Types: application/json

Parameters**Table 10-68 Parameters**

Name	Description
type	Application type
status	-1 - In Progress 0 - Success 1 - Failure
details	In case of errors, details are published with the error string
descriptiveStatus	The status of the job, such as Completed or Error.
items	Collection of notification categories
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/100000000053010",
    }
  ]
}
```

```

        "action": "GET"
    }
],
"status": -1,
"type": "ARCS",
"link": null,
"items": null,
"error": null
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

Removing Users from a Team for Financial Consolidation and Close and Tax Reporting

Removes Oracle Enterprise Performance Management Cloud users listed in a UTF8 formatted CSV file from an existing team in Access Control. The file must be uploaded to the environment before using this API, and the file should be deleted after the API executes. Use the [Upload REST API](#) to upload the file.

A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:

```

User Login, primary_user
jdoe, yes
jane.doe@example.com, no

```

Note: The users are removed only if both these conditions are met:

- User login IDs included in the file exist in the identity domain that services the environment
- The user is assigned to a pre-defined role in the identity domain

The API is asynchronous and returns the Job ID. Use the job status URI to determine whether the process is complete. The presence of status -1 in the response indicates that the update is in progress. Any non-zero status except -1 indicates failure for the update.

Note that this feature uses a Planning REST API to run a job. Details about Planning REST APIs are described here: [Planning REST APIs](#).

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

REST Resource

```

POST /HyperionPlanning/rest/{api_version}/applications/{application}/
fcmjobs

```

Example URL

`https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/applications/FCCS/fcmjobs`

Required Roles

Service Administrators

Request

Supported Media Types: `application/x-www-form-urlencoded`

The following table summarizes the parameters.

Table 10-69 Parameters

Name	Description	Type	Required	Default
<code>api_version</code>	The version of the API you are developing with; for the current release, it is v3	Path	Yes	None
<code>application</code>	The name of the application, FCCS	Path	Yes	None

Example of Request Body

The following table summarizes the parameters of the JSON request.

Table 10-70 Parameters

Name	Description
<code>jobName</code>	The name of the job, REMOVE_USERS_FROM_TEAM
<code>fileName</code>	<p>The name of the uploaded ANSI or UTF-8 encoded CSV file containing information on the users to be removed, for example, <code>removeUsersFromTeam.csv</code>.</p> <p>The file must have been uploaded already using the Upload REST API. The CSV file should not include the account of the user who executes this command.</p> <p>A primary user is, by default, designated to perform the tasks that are assigned to the team. The file format is as follows:</p> <pre>User Login, primary_user jdoe, yes jane.doe@example.com, no</pre>
<code>teamName</code>	The name of an existing team in Access Control, such as <code>Team1</code> .

Example of Request body

```
{
  "jobName":"REMOVE_USERS_FROM_TEAM",
  "parameters":{
```

```

        "fileName":"users.csv",
        "teamName":"Team1"
    }
}

```

Response

Supported Media Types: application/json

Parameters

Table 10-71 Parameters

Name	Description
jobName	REMOVE_USERS_FROM_TEAM
jobID	The ID of the job, such as 100000000114040
status	-1 - In Progress 0 - Success 1 - Failure
details	In case of errors, details are published with the error string
descriptiveStatus	The status of the job, such as Completed or Error
items	Collection of notification categories
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Possible value: self
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```

{
  "jobName":"REMOVE_USERS_FROM_TEAM",
  "jobId":100000000114040,
  "descriptiveStatus":",
  "detail":"In Progress",
  "status":-1,
  "items":null,
  "links":[
    {
      "rel":"self",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/FCCS/fcmjobs/100000000114040",
      "action":"GET"
    }
  ]
}

```

Common Functions

- See [Common Helper Functions for Java](#)
- See [Common Helper Functions for cURL](#)
- See [CSS Common Helper Functions for Groovy](#)

11

Reporting REST APIs

Use the topics in this chapter to run reports with REST APIs for Account Reconciliation, Financial Consolidation and Close, Tax Reporting, and Data Management.

For reports on users with REST APIs, see [User Access Report \(v1\)](#) and [User Access Report \(v2\)](#).

Generate Report for Account Reconciliation

Generates either a single predefined Reconciliation Compliance report, predefined Transaction Matching report or a custom report.

This API is version v1.



Note:

All parameters must be specified for a report.

REST Resource

POST `/arm/rest/fcmapi/{api_version}/report`

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request.

Table 11-1 REPORT

Name	Description	Path	Required
<code>api_version</code>	Version of the API you are working with. This release is v1	Yes	Yes
<code>groupName</code>	The name of the group the report is associated with.	No	Yes

Table 11-1 (Cont.) REPORT

Name	Description	Path	Required
reportName	The name of the report to be generated.	No	Yes
generatedReport FileName	The name specified by the user of the report to be generated. If this parameter is not provided, then the report will get generated with the data for reportName parameter in this table.	No	No
parameters	Each report may have different parameters. Types of parameters: <ul style="list-style-type: none"> Numerical - should be in BigDecimal format. Text - standard string Date - can be in format yyyy-MM-dd for example 2020-10-01. To use the current date, use the value "CURRENT_DATE". Date/Time - can be in format yyyy-MM-dd HH:mm:ss or yyyy-MM-dd'T'HH:mm:ss for example 2020-10-01 13:01:00, 2020-10-01T13:01:00. To use the current date and time, use the value "CURRENT_DATE_TIME". Boolean Users - user login ID List of choices - case insensitive values 	No	No
format	The format of the report (HTML, PDF, XLSX, CSV or CSV2).	No	No (default is PDF)
module	The module within Account Reconciliation: RC (Reconciliation Compliance) or TM (Transaction Matching).	No	No (default is RC)
emails	Comma separated list of email addresses that will receive the report once it's generated.	No	No

Table 11-1 (Cont.) REPORT

Name	Description	Path	Required
runAsync	Generation of report runs asynchronously (true) or synchronously (false). Oracle recommends setting this value to true (async) for larger reports. An example of request body and output is shown.	No	No (default is false)

 **Note:**

If the required parameters, `groupName` or `reportName` are not specified, you receive an error.

For details about `reportName` or parameters see *Generating Predefined Reports in Reconciliation Compliance* or *Generating Predefined Reports in Transaction Matching in Administering Account Reconciliation*.

For details about **Output Format**, see *Generating the Report in Administering Account Reconciliation*.

For details about retrieving job status while running reports, see [Retrieve Job Status for a Report](#).

Example of request body (to be run synchronously)

```
{
  "groupName": "Reconciliation Manager",
  "reportName": "Balance by Account Type",
  "generatedReportFileName": "myReport.pdf",
  "parameters": { "Period": "June 2018", "Currency Bucket": "Entered", "Rate
Type": "Accounting" },
  "format": "PDF",
  "module": "RC",
  "emails": "user1@oracle.com,user2@oracle.com",
  "runAsync": false
}
```

Example of request body (to be run asynchronously for larger reports)

```
{
  "groupName": "Reconciliation Manager",
  "reportName": "Balance by Account Type",
  "generatedReportFileName": "myReport.pdf",
  "parameters": { "Period": "June 2018", "Currency Bucket": "Entered", "Rate
Type": "Accounting" },
  "format": "PDF",
```

```
"module": "RC",
"emails": "user1@oracle.com,user2@oracle.com",
"runAsync": true
}
```

Response

Supported Media Types: application/json

Parameters:

Table 11-2 Parameters

Name	Description
type	The module within Account Reconciliation: RC (Reconciliation Compliance) or TM (Transaction Matching).
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type (self, Job Status). if set to Job Status, you can use the href to get the status of the operation
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following is an example of the response body in JSON format for a Reconciliation Compliance successfully completed report called My Report in pdf format generated synchronously (runAsync=false):

```
{
  "type": "RC",
  "status": 0,
  "details": "myReport.pdf",
  "links" [ {
    "action": "POST",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/fcm/rest/fcmapi/v1/
report",
    "rel": "self"
  },
  {
    "rel": "report-content",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/v1/
applicationsnapshots/myReport.pdf",
    "action": "GET"
  }
]
}
```

The following is an example of the response body in JSON format for a Reconciliation Compliance report generated asynchronously (`runAsync=true`) where the report is "In Process" and you can use the Job ID generated to retrieve the job status. See [Retrieve Job Status for a Report](#):

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/
report",
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/
report/job/TM/100000001001009",
      "action": "GET"
    }
  ],
  "details": "In Process",
  "status": -1,
}
```

Generate Report for Financial Consolidation and Close and Tax Reporting

Generates a report for Financial Consolidation and Close (Task Manager, Supplemental Data, and Enterprise Journal) and Tax Reporting (Task Manager and Supplemental Data).

This API is version v1.



Note:

All parameters must be specified for a report.

REST Resource

POST /HyperionPlanning/rest/fcmapi/{api_version}/report

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 11-3 Report Parameters

Name	Description	Path	Required
api_version	Version of the API you are working with. This release is v1	Yes	Yes
groupName	The name of the group the report is associated with.	No	Yes
reportName	The name of the report to be generated.	No	Yes
generatedReportFileName	The user-specified name of the report to be generated. If this parameter is not provided, then the report will get generated with the data for reportName parameter in this table.	No	No
parameters	Each report may have different parameters. Types of parameters: <ul style="list-style-type: none"> • Numerical - should be in BigDecimal format. • Text - standard string • Date - can be in format yyyy-MM-dd for example 2020-10-01 • Date/Time: can be in format yyyy-MM-dd HH:mm:ss or yyyy-MM-dd'T'HH:mm:ss for example 2020-10-01 13:01:00, 2020-10-01T13:01:00 • Boolean • Users - user login ID • List of choices - case insensitive values 	No	No
format	The format of the report (HTML, PDF, XLSX, or CSV).	No	No (Default is PDF)
module	Module for which the report is created. For Financial Consolidation and Close, use SDM (Supplemental Data Manager) or Task Manager. For Tax Reporting, use SDM (Supplemental Data Manager) or Task Manager.	No	No
emails	Comma separated list of email addresses that will receive the report.	No	No

Table 11-3 (Cont.) Report Parameters

Name	Description	Path	Required
runAsync	Generation of report runs asynchronously (true) or synchronously (false). Oracle recommends setting this value to true (async) for larger reports. An example of request body and output is shown.	No	No (Default is false)

For details about `reportName` or `parameters` see [Using Task Manager and Supplemental Data Manager Reports](#).

For details about **Output Format**, see [Generating the Report](#) .

For details about retrieving job status while running reports, see [Retrieve Job Status for a Report](#).

Example of request body (to be run synchronously)

```
{
  "groupName": "Task Manager",
  "reportName": "Late Tasks",
  "generatedReportFileName": "myReport.pdf",
  "parameters": {"Schedule": "Qtr 2 Close", "Period": "Jun" },
  "format": "PDF",
  "module": "Task Manager",
  "emails": "user1@oracle.com,user2@oracle.com",
  "runAsync": false
}
```

Example of request body (to be run asynchronously for larger reports)

```
{
  "groupName": "Task Manager",
  "reportName": "Late Tasks",
  "generatedReportFileName": "myReport.pdf",
  "parameters": {"Schedule": "Qtr 2 Close", "Period": "Jun" },
  "format": "PDF",
  "module": "Task Manager",
  "emails": "user1@oracle.com,user2@oracle.com",
  "runAsync": true
}
```

Response

Supported Media Types: `application/json`

Parameters:

Table 11-4 Parameters

Name	Description
type	Type of report: SDM (Supplemental Data Management) or FCCS (Task Manager).
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type (self, Job Status). If set to Job Status, you can use the href to get the status of the operation.
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following is an example of the response body in JSON format for a Financial Consolidation and Close report called MyReport in pdf format that was run successfully synchronously (runAsync=false):

```
{
  "links": [
    {
      "rel": "self",
      "href":
      "https://<SERVICE_NAME>-
      <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
      rest/fcmapi/v1/myReport.pdf",
      "action": "POST"
    },
    {
      "rel": "report-content",
      "href":
      "https://<SERVICE_NAME>-
      <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
      rest/fcmapi/v1/myReport.pdf",
      "action": "POST"
      "GET"
    }
  ],
  "details": "MyReport.pdf",
  "type": "FCCS",
  "status": 0,
  "link": null,
  "error": null,
  "items": null
}
```

The following is an example of the response body in JSON format for a Financial Consolidate and Close report generated asynchronously (runAsync=true) where the

report is "In Process" and you can use the Job ID generated to retrieve the job status. See [Retrieve Job Status for a Report](#):

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/HyperionPlanning/rest/fcmapi/v1/report",
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<dcX>.oraclecloud.com/HyperionPlanning/rest/fcmapi/v1/report/job/FCCS/100000001001009",
      "action": "GET"
    }
  ],
  "details": "In Process",
  "status": -1,
  "type": "FCCS",
  "link": null,
  "error": null,
  "items": null
}
```

Generate User Details Report for Account Reconciliation

Generates a User Details report for **Account Reconciliation**. The **User Details** report contains information on the users who have predefined roles in the environment and lists attributes of each user (such as name and email), their status, teams, predefined roles, workflow roles, organizations, groups, and last login timestamps.

You can use the **Download** REST API to download the report after generating it.

REST Resource

POST /arm/rest/fcmapi/{api_version}/rc/export/users

A sample Account Reconciliation Access Control report:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Name	User Login	Status	Teams	Email	Role	Workflow	Preparer	Reviewer	Organizations	Power User	Last Login
2	John Doe	jdoe	Available	Admins	john.doe@example.com	Administrator		No	No			
3	Jane Doe	janedoe	Available		jane.doe@example.com	Power User		No	No			
4	App User1	app1	Available	Team A	app.user1@example.com	User		No	No			

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 11-5 REPORT

Name	Description	Path	Required
api_version	Version of the API you are working with. This release is v1	Yes	Yes
fileName	The name of the report to be generated.	No	Yes
format	The format of the report (CSV or XLS).	No	No (default = CSV)



Note:

For details about retrieving job status while running reports, see [Retrieve Job Status for a Report](#).

Examples of request body

Example 1

```
{
  "fileName": "UserDetails.csv",
  "format": "CSV"
}
```

Example 2

```
{
  "fileName": "UserDetails.csv"
}
```

Example 3

```
{
  "fileName": "UserDetails.xls",
  "format": "xls"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 11-6 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type can be (self, or Job Status). If set to Job Status, you can use the href to get the status of the operation

Examples of Response Body

The following is an example of the response body in JSON format for an Account Reconciliation User Details report completed successfully:

Job Response

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
export/users",
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/job/
42233",
      "action": "GET"
    }
  ],
  "details": "In Process",
  "status": -1,
  "type": "rc",
  "link": {},
  "error": null,
  "items": []
}
```

Job Status Response

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/job/
42233",
```

```

        "action": "GET"
      },
      {
        "rel": "report-content",
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/
11.1.2.3.600/applicationsnapshots/UserDetails.csv/contents",
        "action": "GET"
      }
    ],
    "details": "File UserDetails.csv generated successfully.",
    "status": 0,
    "type": "rc",
    "link": null,
    "error": null,
    "items": null
  }
}

```

Generate User Details Report for Financial Consolidation and Close and Tax Reporting

Generates a User Details report (for Task Manager, Supplemental Data, and Enterprise Journal user assignments) in Financial Consolidation and Close and (for Task Manager and Supplemental Data user assignments) in Tax Reporting. The **User Details** report contains information on the users who have predefined roles in the environment and lists attributes of each user (such as name and email) as well as their status, teams, predefined roles, workflow roles, organizations, groups, and last login timestamps.

You can use the **Download** REST API to download the report after generating it.

This API version is v1.

REST Resource

POST /HyperionPlanning/rest/fcmapi/{api_version}/fcm/export/users

A sample User Details Report:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	User Login	Status	Teams	Email	Role	Workflow Task Assignee	Task Approver	Organization	Last Login	Form Preparer	Form Approver	Form Integrator	Groups	
2	John Doe	john.doe@example.com	Available	Admins	john.doe@example.com	Administrator								Service Administrator	
3	Jane Doe	jane.doe@example.com	Available		jane.doe@example.com	Power User								Power User	
4	userAts	ats_user	Available		example1@example.com	User								User	
5	user2Ats	ats_user2	Available		example2@example.com	user								User	
6	View User	viewUser	Available		view.user@example.com	Viewer								Viewer	

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 11-7 REPORT

Name	Description	Path	Required
api_version	Version of the API you are working with. This release is v1	Yes	Yes
fileName	The name of the report to be generated.	No	Yes
format	The format of the report (CSV or XLS).	No	No (default = CSV)



Note:

For details about retrieving job status while running reports, see [Retrieve Job Status for a Report](#).

Examples of request body

Example 1

```
{
  "fileName": "UserDetails.csv",
  "format": "CSV"
}
```

Example 2

```
{
  "fileName": "UserDetails.csv"
}
```

Example 3

```
{
  "fileName": "UserDetails.xls",
  "format": "xls"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 11-8 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type can be (self, or Job Status). If set to Job Status, you can use the href to get the status of the operation

Example of Response Body

The following is an example of the response body in JSON format for User Details report completed successfully:

Job Response

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/fcmapi/v1/fcm/export/users",
      "action": "POST"
    },
    {
      "rel": "Job Status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/fcmapi/v1/fcm/job/39068",
      "action": "GET"
    }
  ],
  "details": "In Process",
  "status": -1,
  "type": "fcm",
  "link": {},
  "error": null,
  "items": []
}
```

Job Status Response

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/fcmapi/v1/fcm/job/39068",
```

```

        "action": "GET"
      },
      {
        "rel": "report-content",
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/userDetail.xls/contents",
        "action": "GET"
      }
    ],
    "details": "File userDetail.xls generated successfully.",
    "status": 0,
    "type": "fcm",
    "link": null,
    "error": null,
    "items": null
  }
}

```

Retrieve Job Status for a Report

Use this REST API to get the processing state for a report job with a specified ID. Using this REST API requires prerequisites, such as understanding how to use jobs. See [Prerequisites](#). Be sure that you understand how to use jobs as described in [Managing Jobs](#).

Required Roles

Service Administrator, Power User, User, Viewer

GET /arm/rest/fcmapi/{api_version}/job/{module}/{jobIdentifier}

REST Resource

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 11-9 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
module	Module for Account Reconciliation Report (valid values are RC or TM) Valid values for Financial Close & Consolidation or Tax Reporting are SDM or FCCS (Task Manager)	Path	Yes	Yes
jobIdentifier	The ID of the job	Path	Yes	None

Parameters

Parameters

The following table summarizes the response parameters.

Table 11-10 Parameters

Name	Description
status	Status of the job: -1 = in process; 0 = completed (success); 1 = error
details	Details about the job status, such as "Big Report 10.csv" for generation of a report in csv format named Big Report 10
jobID	The ID of the job, such as 224
type	The type of report: RC (Reconciliation Compliance); TM (Transaction Matching); FCCS (Task Manager) or SDM (Supplemental Data Manager)

Supported Media Types: application/json

Examples of Response Body

The following shows an example of the response body for a completed (successful) report:

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/report",
      "action": "POST"
    },
    {
      "rel": "report-content",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/Big+Report+10.csv/contents",
      "action": "GET"
    }
  ],
  "details": "Big Report 10.csv".
  "status": 0,
  "type": "RC",
  "link": null,
  "error": null,
  "items": null
}
```

The following shows an example of the response body for an error occurring during report generation:

```
{
  "links": [
    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-
```



```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/
report/job/TM/1145",
    "action": "GET"
  },
]
"details": "Invalid query attached to the report".
"status": 1,
"type": "RC",
"link": null,
"error": null,
"items": null
}

```

The following shows an example of the response body for a report generation that is in process:

```

{
"links": [
{
    "rel": "self",
    "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/
report/job/TM/1124",
    "action": "GET"
  },
]
"details": "In Process".
"status": -1,
"type": "RC",
"link": null,
"error": null,
"items": null
}

```

Execute Reports for Data Management

The Data Management reporting framework represents a unified solution that incorporates source and target data, templates, and user-defined SQL queries. Templates, created in Oracle Business Intelligence Publisher, consume data in XML format and generate reports dynamically. You can add SQL queries to extract data from tables, or couple them with the report parameters to extend the definition of a standard report. Data Management reports can be generated as PDF, Excel, Word, or HTML output.

Required Roles

Service Administrator, Power User

REST Resource

POST /aif/rest/{api_version}/jobs

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 11-11 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1	Path	Yes	None
jobType	The job type, REPORT	Path	Yes	None
jobName	The name of the report to be executed, such as Dimension Map For POV (Dimension, Cat, Per)	Path	Yes	None
reportFormatType	The file format of the report, pdf, xlsx, html, or excel	Path	Yes	pdf
parameters	Can vary in count and values based on the report	Path	Yes	None
Location	The location of the report, such as Comma_Vision	Path	Yes	None

Example of Request Body

The following shows an example of the request body in JSON format.

```
{
  "jobType": "REPORT",
  "jobName": "Dimension Map For POV (Dimension, Cat, Per)",
  "reportFormatType": "PDF",
  "parameters": {
    "Dimension Name": "ENTITY",
    "Category": "Actual",
    "Period": "Jan15",
    "Location": "Comma_Vision"
  }
}
```

For sample code, see the code samples included in [Running Data Rules](#).

Response

The following table summarizes the response parameters.

Table 11-12 Parameters

Name	Description
jobId	The process ID generated in Data Management for the job, such as 1885
status	The job status, such as RUNNING
logFileName	Log file containing entries for this execution, such as outbox\logs\BESSAPP-DB_1885.log
outputFileName	Name of the output file generated; you can use this name to download the report

Table 11-12 (Cont.) Parameters

Name	Description
processType	Type of process executed, EXECUTE_REPORT
executedBy	Login name of the user used to execute the rule, such as admin
details	Returns the exception stack trace in case of an application error, or null

Supported Media Types: application/json

Parameters

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links":
  [
    0]
  "status": "-1",
  "details": "null",
  "jobId": "1885",
  "jobStatus": "RUNNING",
  "logFileName": "outbox/logs/1885.log",
  "outputFileName": "outbox/reports",
  "processType": "EXECUTE_REPORT",
  "executedBy": "admin"
}
```

For sample code, see the code samples included in [Running Data Rules](#).

12

Data Management REST APIs

Use the Data Management REST APIs to run data rules, run batches, import and export data mapping, import and export Data Management APIs, and to execute reports.

URL Structure for Data Management

URL Structure

Use the following URL structure to access the Data Management REST resources:

```
https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/{api_version}/  
{path}
```

Where:

api_version —API version you are developing with. The current REST API version for Data Management is V1.

path —Identifies the resource

Getting API Versions for Data Management APIs

You can manage versions using the set of REST resources summarized in the following table.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 12-1 Manage Versions of Data Management APIs

Task	Request	REST Resource
Get API Versions for Data Management APIs	GET	/aif/rest/
Get Information about a Specific API Version for Data Management APIs	GET	/aif/rest/{apiVersion}

Get API Versions for Data Management APIs

Returns information about which versions are available and supported. Multiple versions might be supported simultaneously.

**Note:**

An API version is always supported even when deprecated.

REST Resource

GET /aif/rest/

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 12-2 Parameters

Name	Description
items	Detailed information about the API
version	The version, such as V1
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [1]
  {
    "version": "V1"
    "isLatest": "true"
    "lifecycle": "active"
    "links": [3]
    {
      "rel": "self"
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/"
      "action": "GET"
    }, {
      "rel": "canonical"
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/"
```

```

    "action": "GET"
  }, {
    "rel": "current"
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/V1"
    "action": "GET"
  }
}
}

```

Get Information about a Specific API Version for Data Management APIs

Returns details for a specific REST API version for Data Management.

REST Resource

GET /aif/rest/{api_version}

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 12-3 Parameters

Name	Description
api_version	Version of the API you are developing with, such as V1

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 12-4 Parameters

Name	Description
version	The version, such as V1
lifecycle	Lifecycle of the resource, active or deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```

{
  "version": "V1"
  "lifecycle": "active"
  "isLatest": "true"
  "links": [1]{
    "rel": "canonical"
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/V1"
    "action": "GET"
  }
}

```

Running Data Rules

Executes a Data Management data load rule based on the start period and end period, and import or export options that you specify.

Prerequisites

- Data Rules: Data load rules define how Integrations load data from a file. You must have predefined data load rules to load data.
- You must have the required privileges to execute a specific data rule.

REST Resource

POST /aif/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 12-5 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1	Path	Yes	None
jobType	should be set to "DATARULE"		Yes	None
jobName	The name of a data load rule defined in Data Management. You should enclose the rule name in quotation marks if it contains a space.		Yes	None

Table 12-5 (Cont.) Parameters

Name	Description	Type	Required	Default
startPeriod	The first period for which data is to be loaded. This period name must be defined in Data Management period mapping.		Yes	None
endPeriod	The last period for which data is to be loaded. This period name must be defined in Data Management period mapping.		Yes	None
importMode	Determines how the data is imported into Data Management. Acceptable values are: <ul style="list-style-type: none">• APPEND to add to the existing POV data in Data Management• REPLACE to delete the POV data and replace it with the data from the file.• RECALCULATE to skip importing the data, but re-process the data with updated Mappings and Logic Accounts.• NONE to skip data import into Data Management staging table		Yes	None

Table 12-5 (Cont.) Parameters

Name	Description	Type	Required	Default
exportMode	<p>Determines how the data is exported into Data Management.</p> <p>Acceptable values for Planning Modules and Planning are:</p> <ul style="list-style-type: none"> • STORE_DATA to merge the data in the Data Management staging table with the existing Planning data • ADD_DATA to add the data in the Data Management staging table to Planning • SUBTRACT_DATA to subtract the data in the Data Management staging table from existing Planning data • REPLACE_DATA to clear the POV data and replace it with data in the Data Management staging table. The data is cleared for Scenario, Version, Year, Period, and Entity • NONE to skip data export from Data Management to Planning <p>Acceptable values for Financial Consolidation and Close and Tax Reporting are:</p> <ul style="list-style-type: none"> • REPLACE to delete the POV data and replace it with the data from the file • MERGE: By default, all data load is processed in the Merge mode. If data already existed in the application, the system overwrites the existing data with the new data from the load file. If data does not exist, the new data will be created. • NONE to skip the data export 		Yes	None
fileName	<p>An optional file name. If you do not specify a file name, this API imports the data contained in the file name specified in the load data rule. The data file must already reside in the Inbox prior to data rule execution.</p> <p>Import data files from the EPM INBOX accessible from the Applications-Inbox/Outbox Explorer using a file name. Reference the files in this folder using #epminbox/<filename>.</p>		Yes	None

Example URL

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcx>.oraclecloud.com/aif/rest/v1/jobs
```

Example of Request Body

```
{
  "jobType": "DATARULE",
  "jobName": "aso to bso dr",
  "startPeriod": "Dec-18",
  "endPeriod": "Dec-18",
  "importMode": "REPLACE",
  "exportMode": "NONE",
  "fileName": "#epminbox/TestData.txt"
}
```

Response

Supported Media Types: application/json

Table 12-6 Parameters

Name	Description
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 = cancel pending; 3 = cancelled; 4 = invalid parameter
jobStatus	A text representation of the job status, with one of the following values "RUNNING", "SUCCESS", "FAILED"
jobId	The process ID generated in Data Management for the job
logFileName	Log File containing entries for this execution.
outputFileName	Name of the output file generated, if any.
processType	Type of the process executed. Will contain "COMM_LOAD_BALANCES" for all Data Rule executions
executedBy	Login name of the user used to execute the rule.
details	Returns the exception stack trace in case of an application error

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "jobStatus": "RUNNING"
  "jobId": 2019
  "logFileName": "\\outbox\logs\Account Reconciliation Manager_2019.log"
  "outputFileName": null
  "processType": "COMM_LOAD_BALANCES"
  "executedBy": "admin"
  "status": -1
  "links": [1]
    0: {
      "rel": "self"
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/v1/jobs/2019"
      "action": "GET"
    }
  "details": null
}
```

}

Lock and Unlock POV

The lock REST API prevents data from being loaded to a selected POV by location or application associated with a current period and category.

When a location or application has been locked, you cannot import, validate, export, or rerun a validation.

An unlock RESP API is available so that you can unlock a locked location or application.

Prerequisites:

You must have Service Administrator privileges to execute the lock and unlock REST APIs.

REST Resource

`/aif/rest/V1/POV`

Required Roles

Service Administrator, Power User

Method:

POST

Lock/Unlock by Location RESP API Request Parameters

The following table summarizes the request parameters for locking by location:

Table 12-7 Parameters

Name	Description	Type	Required	Default
api_version	V1	Path	Yes	None
category	Specify the predefined Scenario value based on the POV Category from the integration (data rule) definition. The categories available are those that are created in the Data Management set-up, such as "Actual."	JSON payload	Yes	None
period	Specify the period of the POV from the integration or data load rule defined in Data Management.	JSON payload	Yes	None
location	Specify the name of the location to lock so that data cannot be loaded to it.	JSON payload	Yes	None
locktype	Specify location for the location lock type.	JSON payload	Yes	None

Table 12-7 (Cont.) Parameters

Name	Description	Type	Required	Default
operation	For a lock operation, specify lock . For an unlock operation, specify unlock .	JSON payload	Yes	None

JSON Request Payload to Lock Location Example:

```
{
  "category": "Actual",
  "period": "Jan-17",
  "location": "FCCSAPP1_LOC1A",
  "locktype": "location",
  "operation": "lock"
}
```

Response to Locking a Location Example:

```
{
  "details": "",
  "status": 0,
  "response": "Location:FCCSAPP1_LOC1A has been locked successfully."
}
```

JSON Request Payload to Unlock Location Example:

```
{
  "category": "Actual",
  "period": "Jan-17",
  "location": "FCCSAPP1_LOC1A",
  "locktype": "location",
  "operation": "unlock"
}
```

Response to Unlocking a Location Example:

The following is an example of unlocking a location REST API response.

```
{
  "details": "",
  "status": 0,
  "response": "Location:FCCSAPP1_LOC1A has been unlocked successfully."
}
```

Lock/Unlock by Application REST API Request Parameters

Name	Description	Type	Required	Default
api_version	V1	Path	Yes	None

Name	Description	Type	Required	Default
category	Specify the predefined "Scenario" value based on the POV Category from the integration (data rule) definition. The categories available are those that were created in the Data Management set-up, such as "Actual".	JSON payload	Yes	None
period	Specify the period of the POV from the integration or data load rule defined in Data Management.	JSON payload	Yes	None
application	Specify the name of the application to lock so that data cannot be loaded to it.	JSON payload	Yes	None
locktype	Specify application for the application lock type.	JSON payload	Yes	None
unlockbylocation	true/false—Boolean option for whether to unlock by location when an application is locked. If the "unlockbylocation" flag is set to "true" when locking the target application, then the system locks all rules present in the location under target application and not the application level lock. If the "unlockbylocation" flag is set to "false" when locking the target application, then the system locks all rules present in the location under the target application and the application level lock. Rules present for locations in the application cannot be executed when the location is unlocked until the application level lock is removed. Furthermore, when you create a new location under the locked target application, rules can't be executed in the new location until the application level lock is removed.	JSON payload	No	false
operation	For a lock operation, specify lock . For an unlock operation, specify unlock .	JSON payload	Yes	None

JSON Request Payload to Lock an Application Example:

```
{
  "category": "Actual",
  "period": "Jan-17",
  "application": "FCCSAPP1",
  "locktype": "application",
```

```
"operation": "lock"
}
```

Response to Locking an Application Example:

The following is an example of locking an application REST API response.

```
{
  "details":null,
  "status": 0,
  "response":"Application: FCCSAPP1 has been locked successfully."
}
```

JSON Request Payload to Unlock a Location for a Locked Application Example:

```
{
  "category": "Actual",
  "period": "Jan-17",
  "application": "FCCSAPP1",
  "locktype":"application",
  "unlockbylocation":"true",
  "operation": "lock"
}
```

Response to Unlocking a Location for a Locked Application Example:

The following is an example of unlocking a location when locking an application REST API response.

```
{
  "details":null,
  "status": 0,
  "response":"Application: FCCSAPP1 has been locked successfully."
}
```

Integration Job Type

Use the INTEGRATION job type for the jobs REST API to execute an integration or data load rule based on how periods are processed and source filters.

The INTEGRATION job type is an enhanced version of DATARULE job type (see [Running Data Rules](#)). It is recommended that you use the INTEGRATION job type for future integration jobs.

The INTEGRATION jobtype supports running integrations/data load rules based on:

- period names provided to Planning
- Global POVs
- Planning substitution variables
- source filters selected as runtime parameters
- target options selected as runtime parameters

- existing period ranges

It also supports overriding the source filters and target options at runtime without modifying the integration definition.

Prerequisites:

You must have the required privileges to execute a specific data rule/integration.

REST Resource

```
/aif/rest/{api_version}/jobs
```

Required Roles

Service Administrator, Power User

Method:

POST

Payload:

```
{
    "jobType": "INTEGRATION",
    "jobName": "GLDATA",
    "periodName": "{Mar-20}",
    "importMode": "Replace",
    "exportMode": "Merge",
    "fileName": "inbox/GLBALANCE.txt"
}
```

REST Payload Description

The following table summarizes the REST payload.

Table 12-8 Parameters

Name	Description	Type	Required	Default
api_version	V1	Path	Yes	None
jobType	INTEGRATION	JSON payload	Yes	None
jobName	The name of the integration or data load rule defined in Data Management. You should enclose the rule or name in quotation marks if it contains a space.	JSON payload	Yes	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
periodname	<p>Name of the period(s)p enclosed in curly brackets ({}).</p> <p>periodname includes:</p> <ul style="list-style-type: none"> • Single Period—Refers to the Data Management period name for a single period. The parameter is whatever the period name is defined in Period mapping. • Multi-Period—Refers to a multi-period load. The parameter is {Month-Year}{Month-Year}. For example, {Jan-20}{Mar-20} refers to a multi-period load from Jan-20 to Mar-20. • Planning Period Name—Refers to a Planning period name. The parameter is {Month#Year}, for example, {Jan#FY20}{Mar#FY20}. Using this convention, the client executing the API does not need to know the Data Management period names. Instead you specify the Planning member names for the Year and Scenario dimensions. <p>This parameter is supported in the Planning, Tax Reporting, and Financial Consolidation and Close business processes. It is functional for both your service applications and cloud deployments derived from on-premises data sources.</p> <p>This parameter is useful if triggered from an Oracle Enterprise Performance Management Cloud Groovy script by capturing the Year and Period member names. The application period mapping or global period mapping must exist with the Year and Month in the target values of the period mapping.</p> <ul style="list-style-type: none"> • Planning Substitution Variable—This is an extension of the previous Planning period name parameter whereby a substitution variable can be specified instead of the actual Year/Month member names. <p>The convention is {Month#&CurYr} {&FcstMonth#&CurYr}; for example, {Jan#&CurYr} {&FcstMonth#&CurYr}.</p>	JSON payload	Yes	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
	<p>A combination of both actual member names as well as substitution variables is supported.</p> <p>This parameter is supported in the Planning, Tax Reporting, and Financial Consolidation and Close business processes. It is functional for both your service applications and cloud deployments derived from on-premises data sources.</p> <p>The application period mapping or global period mapping must exist in the Data Management of the instance where the API is executed, with the Year and Month in the target values of the period mapping. In this case, Year and Month refer to the current value of the substitution variable during execution.</p> <ul style="list-style-type: none"> • GLOBAL POV—Executes the data load for the Global POV period. Use the format {GLOBAL_POV}. 			

 **Note:**

If you use any other period naming parameter other than the parameters described above, you get an "Invalid Input – HTTP 400 " error message.

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
importMode	<p>Determines how the data is imported into Data Management.</p> <p>Acceptable values are:</p> <ul style="list-style-type: none">• Append—Add to the existing POV data in Data Management• Replace—Delete the POV data and replace it with the data from the file.• Map and Validate—Skip importing the data, but re-process the data with updated Mappings and Logic Accounts.• No Import—Skip data import into Data Management staging table.• Direct—To use the direct load method to extract data from your on-premises data sources and then load the data directly to the EPM Cloud using the EPM Integration Agent, you need to pass an importMode of "Direct". Other modes are not applicable for the direct load.	JSON payload	Yes	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
exportMode	<p>Determines how the data is exported into Data Management.</p> <p>Acceptable values for Planning business processes are:</p> <ul style="list-style-type: none"> • Merge—Merge the data in the Data Management staging table with the existing Planning data. • Replace—Clear the POV data and replace it with data in the Data Management staging table. The data is cleared for Scenario, Version, Year, Period, and Entity dimensions. • Accumulate—Add the data in the Data Management staging table to Planning. • Subtract—Subtract the data in the Data Management staging table from existing Data Management data. • No Export—Skip data export from Data Management to Planning. <p>Acceptable values for Financial Consolidation and Close and Tax Reporting are:</p> <ul style="list-style-type: none"> • Merge—Merge the data in the staging table with the data in the Financial Consolidation and Close and Tax Reporting application. If data already exists in the application, the system overwrites the existing data with the new data from the load file. If data does not exist, the new data is created. • Replace—Delete the POV data and replace it with the data from the file. • No Export—Skip the data export from Data Management to Financial Consolidation and Close or Tax Reporting <p>If you use a regular data load for the Numeric Data Only and Numeric Data and All Data Type load methods, note the export mode requirements based on the load method:</p> <ul style="list-style-type: none"> • For a Numeric Data Only load method, Accumulate and Subtract are the applicable export modes. • For a Numeric Data and All Data Type load method, Merge, Replace, and No Export are the applicable export modes. <p>If you use the direct data load to extract data from your on-premises data</p>	JSON payload	Yes	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
	<p>sources and then load the data directly to the EPM Cloud using the EPM Integration Agent, note the export mode requirements based on the load method:</p> <ul style="list-style-type: none"> For a Numeric Data Only load method, Merge, Accumulate, and Subtract are the applicable export modes. For a Numeric Data and All Data Type load method, only Replace is the applicable export mode. The No Export mode is not applicable for the Numeric Data Only and Numeric Data and All Data Type load methods. 			
fileName	<p>The <code>fileName</code> parameter is applicable only for native file-based data loads and ignored if specified for other loads.</p> <p>The file name is optional. If you do not specify a file name, this API imports the data contained in the file name specified in the load data rule. The data file must already reside in the Inbox prior to executing the data load rule, for example, <code>.inbox/GLBALANCES.txt</code>.</p> <p>You can also upload file to folders accessible from the Applications-Inbox/Outbox Explorer using a file name. Reference the files in this folder using this format: <code>#epminbox/<filename></code>.</p>	JSON payload	No	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
sourceFilters	<p>A parameter used to update the source filters defined for the data load rule or integration.</p> <p>File-based applications—You cannot use the sourceFilters parameter for file-based applications. If you use this parameter with a file-based application, you get the HTTP 400 error message: "EPMFDM-ERROR: Data Load Rule does not support sourceFilters for File based loads."</p> <p>Data Source based applications—Replaces the pre-defined source filter parameters at the rule level for a data load rule/integration or specifies them at runtime (if not pre-defined at application level when the data source is the type of source application.</p> <p>Each filter name and its value should be sent as a key/value pair in the nested JSON object. The parameter name and value should be the English display names as seen in the user interface. Do not specify internal codes for parameter names and values using LOV validation. LOV validation is done for parameters having a restricted list of parameter values.</p> <p>All filter names are not mandatory. Only filter names specified in the nested JSON object are replaced or set at runtime. The remaining filters are picked from the application/rule definition.</p> <p>Essbase/Planning/Oracle General Ledger-based applications—Replaces the already defined dimension source filters for a data load rule/integration at runtime when Essbase/Planning/Oracle General Ledger balance type are the source applications.</p> <p>Each dimension name and its value should be sent as a key/value pair in the nested JSON object.</p> <p>Replacing or setting the dimension filters at run time is only supported for dimensions already having a filter defined in the data load rule/integration definition.</p> <p>All filter names are not mandatory. Only dimension names specified in the nested JSON object will be replaced or set in runtime. The remaining filters are</p>	JSON payload	No	None

Table 12-8 (Cont.) Parameters

Name	Description	Type	Required	Default
targetOptions	<p>picked from data rule/integration definition.</p> <p>Dimension filters are supported for all dimensions including the "Scenario" dimension. The "Year" and "Period" dimensions are not supported as filters because they are driven by the POV range.</p> <p>All other source applications are not supported.</p> <p>A parameter used to update the target options defined for the data load rule or integration.</p> <p>Data Export based applications— Replaces the pre-defined target option parameters at the rule level for a data load rule/integration or specifies them at runtime (if not pre-defined at the rule level) when the Data Export is the type of target application.</p> <p>Each option name and its value should be sent as a key/value pair in the nested JSON object. The parameter name and value should be the English display names as seen in the user interface. Do not specify internal codes for parameter names and values using the LOV validation. LOV validation is done for parameters having a restricted list of parameter values.</p> <p>All option names are not required. Only option names specified in the nested JSON object are replaced or set at runtime. The remaining options are picked from the application/rule definition.</p> <p>Planning target application—Replaces the following options at runtime.</p> <ul style="list-style-type: none"> • Refresh Database—Yes/No • Dimension Name—Specify a dimension name for a custom dimension load rule. In this way the same rule can be used to build multiple dimensions in runtime. • Purge Data File—Yes/No <p>Other target applications are not supported.</p>	JSON payload	No	None

Sample REST Payloads

Below are sample payloads based on the source application type.

File Based Loads

In this example, the source is a text file running an integration through a native File adapter:

```
{
  "jobType": "INTEGRATION",
  "jobName": "ERPDATA",
  "periodName": "{Jan-20}",
  "importMode": "REPLACE",
  "exportMode": "NONE",
  "fileName": "inbox/TestData.txt"
}
```

Planning Applications

In these examples, the source is an Essbase/Planning based application. The supported applications include:

- Planning modules
- Reporting cubes (plan types) of Planning
- Financial Consolidation and Close
- Tax Reporting
- Profitability and Cost Management
- Oracle ERP Cloud - Oracle General Ledger Balances Cube

Example of Planning to Financial Consolidation and Close Data Synchronization

```
{
  "jobType": "INTEGRATION",
  "jobName": "PBCStoFCCS",
  "periodName": "{Jan-20}",
  "importMode": "REPLACE",
  "exportMode": "NONE",
  "sourceFilters": {
    "Account": "@RELATIVE (Acc1, 0)",
    "Entity": "@CHILDREN (Europe)",
    "Scenario": "Actual"
  }
}
```

Example of Planning to Data Export to File

```
{
  "jobType": "INTEGRATION",
  "jobName": "PBCStoFCCS",
  "periodName": "{Jan-20}",
  "importMode": "REPLACE",
  "exportMode": "NONE",
  "sourceFilters": {
    "Account": "@RELATIVE (Acc1, 0)",
    "Entity": "@CHILDREN (Europe)",
    "Scenario": "Actual"
  },
}
```

```
"targetOptions":{
  "Download File Name":"PlanningToFile.csv",
  "Column Delimiter":",",
  "Include Header":"Yes"
}
}
```

Data Source Applications

Example of Incremental File adapter:

```
{
"jobType":"INTEGRATION",
"jobName":"MyIncrementalFileLoad",
"periodName":"{Jan-20}{Mar-20}",
"importMode":"REPLACE",
"exportMode":"NONE",
"sourceFilters":{"Source File":"File1.txt"}
}
```

Example of Netsuite adapter:

```
{
"jobType":"INTEGRATION",
"jobName":"NetsuiteLoad",
"periodName":"{Jan-20}{Mar-20}",
"importMode":"REPLACE",
"exportMode":"NONE",
"sourceFilters":{
  "Postingperiod":"This Fiscal Quarter",
  "Mainline":"True"
}
}
```

Example of exporting Oracle NetSuite adapter toPlanning dimensions (metadata rule):

```
{
"jobType":"INTEGRATION",
"jobName":"NetsuiteMetadataLoad",
"periodName":"{Jan-20}{Mar-20}",
"importMode":"REPLACE",
"exportMode":"NONE",
"targetOptions":{
  "Refresh Database":"Yes",
  "Dimension Name":"Product"
}
}
```

Example of Oracle ERP Cloud (Payables Transactions):

```
{
"jobType":"INTEGRATION",
"jobName":"Payables1Load",
```



```

"periodName": "{Jan-20}",
"importMode": "REPLACE",
"exportMode": "NONE",
"sourceFilters": {
  "Invoice Type": "Credit Memo",
  "Cancelled Invoices Only ": "Yes"
}
}

```

Example of Direct Load:

```

{
{
  "jobType": "INTEGRATION",
  "jobName": "DirectLoad_LOCl_DL1",
  "periodName": "{Jan-17}",
  "importMode": "Direct",
  "exportMode": "Merge",
  "executionMode": "ASYNC"
}
}

```

Import Data Mapping

Member mappings are used to derive the target members for each dimension based on source value. Member mappings are referenced during the data load, enabling Data Management to determine how to dimensionalize the data that is loaded to the target application. Member mappings define relationships between source members and target dimension members within a single dimension. You must create a member mapping for each target dimension.

You can import member mappings from a selected Excel, .CSV or .TXT file. You can also create new mappings in a text file and import them. Import member mappings support merge or replace modes, along with validate or no validate options for target members.

REST Resource

POST /aif/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 12-9 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1	Path	Yes	None

Table 12-9 (Cont.) Parameters

Name	Description	Type	Required	Default
jobType	The job type, MAPPINGIMPORT	Path	Yes	None
jobName	The dimension name for a specific dimension to import, such as ACCOUNT, or ALL to import all dimensions	Path	Yes	None
fileName	The file and path from which to import mappings. The file format can be .CSV, .TXT, .XLS, or .XLSX. The file must be uploaded prior to importing, either to the inbox or to a sub-directory of the inbox. Include the inbox in the file path, for example, inbox/BESSAPPJan-06.csv	Path	Yes	None
importMode	The import mode: MERGE to add new rules or replace existing rules, or REPLACE to clear prior mapping rules before import	Path	No	MERGE
validationMode	Whether to use validation mode, true or false An entry of true validates the target members against the target application; false loads the mapping file without any validations. Note that the validation process is resource intensive and takes longer than the validation mode of false; the option selected by most customers is false	Path	No	false
locationName	The Data Management location where the mapping rules should be loaded; mapping rules are specific to a location in Data Management	Path	No	None

Example of Request Body

The following shows an example of the request body in JSON format.

```
{
  "jobType": "MAPPINGIMPORT",
  "jobName": "ACCOUNT"
  "fileName": "inbox/BESSAPPJan-06.csv",
  "importMode": "MERGE",
  "validationMode": "false",
  "locationName": "BESSAPP"
}
```

For sample code, see the code samples included in [Running Data Rules](#).

Response

The following table summarizes the response parameters.

Table 12-10 Parameters

Name	Description
jobId	The process ID generated in Data Management for the job, such as 1880
jobStatus	The job status, such as <code>RUNNING</code>
logFileName	Log file containing entries for this execution, such as <code>outbox/logs/BESSAPP-DB_1880.log</code>
outputFileName	Name of the output file generated, if any, or else <code>null</code>
processType	Type of process executed, <code>IMPORT_MAPPING</code>
executedBy	Login name of the user used to execute the rule, such as <code>admin</code>
details	Returns the exception stack trace in case of an application error, or <code>null</code>

Supported Media Types: `application/json`

Parameters

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links":
  [
    0]
  "status": "-1"
  "details": "null"
  "jobId": "1880"
  "jobStatus": "RUNNING",
  "logFileName": "outbox/logs/BESSAPP-DB_1880.log",
  "outputFileName": "null",
  "processType": "IMPORT_MAPPING",
  "executedBy": "admin"
}
```

For sample code, see the code samples included in [Running Data Rules](#).

Running Batch Rules

Executes a batch of jobs that have been defined in Data Management .

Prerequisites

- The batch must be defined in Data Management before it can be executed using the REST API.
- You must have the required privileges to execute a specific batch.

REST Resource

POST /aif/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 12-11 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1	Path	Yes	None
jobType	should be set to "BATCH"		Yes	None
jobName	The name of a batch defined in Data Management.		Yes	None

Example URL

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/V1/jobs`

Example of Request Body

```
{ "jobType": "BATCH",
  "jobName": "BatchDataLoad"
}
```

Response

The following table summarizes the response parameters.

Table 12-12 Parameters

Name	Description
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 = cancel pending; 3 = cancelled; 4 = invalid parameter
jobStatus	A text representation of the job status, with one of the following values "RUNNING", "SUCCESS", "FAILED"
jobId	The process Id generated in Data Management for the job
logFileName	Log File containing entries for this execution.
outputFileName	Name of the output file generated, if any.
processType	Type of the process executed. Will contain "COMM_BATCH" for all Data Rule executions
executedBy	Login name of the user used to execute the rule.
details	Returns the exception stack trace in case of an application error

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body in JSON format.

```

{
  "jobStatus": "SUCCESS"
"jobId": 2016
"logFileName": "\outbox\logs\BATCH1_7595.log"
"outputFileName": null
"processType": "COMM_BATCH"
"executedBy": "admin"
"status": -1
"links": [1]
  0: {
    "rel": "self"
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/V1/jobs/
2016"
    "action": "GET"
  }
"details": null
}

```

For sample code, see the code samples included in [Running Data Rules](#).

Export Data Mapping

Member mappings are used to derive the target members for each dimension based on source value. Member mappings are referenced during the data load, enabling Data Management to determine how to dimensionalize the data that is loaded to the target application. Member mappings define relationships between source members and target dimension members within a single dimension. You must create a member mapping for each target dimension.

You can export member mappings to a selected file of format .csv, .txt, .xls, or .xlsx.

REST Resource

POST /aif/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 12-13 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1	Path	Yes	None

Table 12-13 (Cont.) Parameters

Name	Description	Type	Required	Default
jobType	The job type, MAPPINGEXPORT	Path	Yes	None
jobName	The dimension name for a specific dimension to import, such as ACCOUNT, or ALL to import all dimensions	Path	Yes	None
fileName	The file and path from which to export mappings. The file format can be .CSV, .TXT, .XLS, or .XLSX. Include the outbox in the file path, for example, outbox/BESSAPPJan-06.csv	Path	Yes	None
locationName	The name of the location, such as BESSAPP	Path	Yes	None

Example of Request Body

The following shows an example of the request body in JSON format.

```
{
  "jobType": "MAPPINGEXPORT",
  "jobName": "ACCOUNT",
  "fileName": "outbox/BESSAPPJan-06.csv",
  "locationName": "BESSAPP"
}
```

For sample code, see the code samples included in [Running Data Rules](#).

Response

The following table summarizes the response parameters.

Table 12-14 Parameters

Name	Description
jobId	The process ID generated in Data Management for the job, such as 1881
jobStatus	The job status, such as SUCCESS
logFileName	Log file containing entries for this execution, such as outbox/logs/BESSAPP-DB_1881.log
outputFileName	Name of the output file generated, such as outbox/BESSAPPJan-06.csv
processType	The type of process executed, EXPORT_MAPPING
executedBy	Login name of the user used to execute the rule, such as admin
details	Returns the exception stack trace in case of an application error, or else null

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "links":
  [
  0]
  "status": "0",
  "details": "null",
  "jobId": "1881",
  "jobStatus": "SUCCESS",
  "logFileName": "outbox/logs/BESSAPP-DB_1881.log",
  "outputFileName": "outbox/BESSAPPJan-06.csv",
  "processType": "EXPORT_MAPPING",
  "executedBy": "admin"
}
```

For sample code, see the code samples included in [Running Data Rules](#).

Export Data Management

The Export Data Management API enables you to back up setup and staging data in Data Management as a snapshot.

REST Resource

/aif/rest/v1/snapshots

Required Roles

Service Administrator, Power User

Method

POST

Request

Supported Media Types: application/json

Sample REST API Payload for Export Data Management

```
{
  "action": "EXPORT",
  "snapshotType": "ALL",
  "fileName": "MyBackup.zip",
  "overwriteFile": true
}
```

The following table summarizes the client request.

Table 12-15 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V1.	Path	Yes	None
action	EXPORT	Payload	Yes	None
snapshottype	Snapshot type: ALL, ALL_INCREMENTAL, INCREMENTAL, SETUP <ul style="list-style-type: none"> • ALL—Include all setup and staging data. • ALL_INCREMENTAL—Include only new or changed staging data based on the POV since the last snapshot was exported and include SETUP and all POVs (old and new) in the output file. • INCREMENTAL—Include only new or changed staging data based on the POV since the last snapshot was exported and include only SETUP and new POVs in the output file. • SETUP—Include only setup data. 	Payload	Yes	None
fileName	Name of the output file in ZIP format. This file is generated in the outbox as: outbox/<filename>.zip. If the file doesn't end with a ZIP extension, Data Management appends the ZIP file extension at the end of the file name.	Payload	Yes	None
overwriteFile	true/false—Boolean option to specify whether or not to replace the output file specified in the filename parameter. This parameter prevents a user from accidentally overwriting the output file if it already exists by throwing a HTTP 400 error.	Payload	No	false

Response

The following table summarizes the response parameters.

Table 12-16 Parameters

Name	Description
action	Always EXPORT
snapshotType	Name of the snapshot type

Table 12-16 (Cont.) Parameters

Name	Description
jobId	The process ID generated in Data Management for the job, such as 1880
links	Describes links to other resources and actions applicable on the current resource.
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 =cancel pending; 3 = cancelled; 4 = invalid parameter;Integer.MAX_VALUE = unknown

Supported Media Types: application/json

The following shows an example of the response in JSON format.

```
{
  "action": "EXPORT",
  "snapshotType": "SETUP",
  "jobId": 423,
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/v1/jobs/
jobID",
      "action": "GET"
    }
  ],
  "status": -1
}
```

Import Data Management

The Import Data Management API enables you to restore setup and staging data from one environment to another. The system clears the existing data in the target environment and then imports the data from the backup files without merging any operations.

REST Resource

/aif/rest/v1/snapshots

Required Roles

Service Administrator, Power User

Method

POST

Request

Supported Media Types: application/json

Sample REST API Payload for Import Data Management

```
{
  "action": "IMPORT",
  "fileName": "inbox/MyBackup.zip"
}
```

The following table summarizes the client request.

Table 12-17 Parameters

Name	Description	Type	Required	Default
action	IMPORT	Payload	Yes	None
filename	File name of the import snapshot (for example: inbox/<filename>.zip, or inbox/mybackup/<filename>.zip). If no path is specified, it is assumed that the file is in the Data Management root folder. (Files to the Data Management root folder can only be uploaded from the Data Management user interface.)	Payload	Yes	None

Response

The following table summarizes the response parameters.

Table 12-18 Parameters

Name	Description
action	Always IMPORT
jobId	The process ID generated in Data Management for an import Data Management job is "0."
links	Describes links to other resources and actions applicable on the current resource.
status	Status of the job: -1 = in progress; 0 = success; 1 = error; 2 =cancel pending; 3 = cancelled; 4 = invalid parameter; Integer.MAX_VALUE = unknown

Supported Media Types: application/json

The following shows an example of the response in JSON format.

```
{
  "action": "IMPORT",
  "jobId": 0,
  "links": [
```

```
        {
            "rel": "self",
            "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/aif/rest/v1/jobs/
jobID",
            "action": "GET"
        }
    ],
    "status": -1
}
```

13

Account Reconciliation APIs

Use the Account Reconciliation REST APIs to get the REST API version, create reconciliations, change period status, import pre-mapped transactions, import profiles, import currency rates, import balances, import pre-mapped balances, monitor reconciliations, and retrieve job status. In Transaction Matching, you can use REST APIs to import pre-mapped transactions, or run auto match.

URL Structure for Account Reconciliation

Use the following URL structure to access the Account Reconciliation REST resources:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
armARCS/rest/{api_version}/{path}
```

Where:

api_version —API version you are developing with. The current REST API version for Account Reconciliation is V1.

path —Identifies the resource

Getting API Versions for Account Reconciliation REST APIs

You can manage versions using the set of REST resources summarized in the following table.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 13-1 Manage Versions of Account Reconciliation APIs

Task	Request	REST Resource
Get API Versions for Account Reconciliation REST APIs	GET	/armARCS/rest/
Get Information about a Specific API Version for Account Reconciliation REST APIs	GET	/armARCS/rest/{apiVersion}

Get API Versions for Account Reconciliation REST APIs

Returns information about which versions are available and supported. Multiple versions might be supported simultaneously.

**Note:**

An API version is always supported even when deprecated.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /armARCS/rest/

Request

Supported Media Types: application/json

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 13-2 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
items	Version of the API you are developing with
version	The version, such as v1
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self or Job Status. If set to Job Status, you can use the href to get the status of the import operation
data	The parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "isLatest": false,
    "lifecycle": "deprecated",
```

```

    "version": "v1",
    "links": [{
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
      "rel": "canonical"
    }, {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
      "rel": "successor-version"
    }
  ], {
    "isLatest": true,
    "lifecycle": "active",
    "version": "v1",
    "links": [{
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
      "rel": "canonical"
    }, {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
      "rel": "predecessor-version"
    }
  ]
}], {
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
    "rel": "canonical"
  }, {
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
    "rel": "current"
  }
]}
}

```

Get Information about a Specific API Version for Account Reconciliation REST APIs

Returns details for a specific REST API version for Account Reconciliation.

REST Resource

GET /armARCS/rest/{api_version}

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 13-3 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as V1	Path	Yes	None

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 13-4 Parameters

Name	Description
version	The version, such as v1
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	The parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "version": "v1",
  "lifecycle": "active",
  "isLatest": true,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
  }, {
    "rel": "predecessor-version",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1",
  }]
}
```

Execute a Job in Account Reconciliation

Use this resource to execute the job by providing the job name and type. The job is expected to be defined in Account Reconciliation with all the required parameters saved with the job definition. For some job types, the parameters can be either provided or overwritten at runtime.

Reconciliation Compliance Supported Job Types:

- CREATE_RECONCILIATIONS
- SET_PERIOD_STATUS
- IMPORT_PREMAPPED_TRANSACTIONS
- IMPORT_PREMAPPED_PROFILES
- IMPORT_RATES
- IMPORT_BALANCES
- MONITOR_RECONCILIATIONS
- IMPORT_PREMAPPED_BALANCES

Transaction Matching Supported Job Types:

- IMPORTTMPREMAPPEDTRANSACTIONS
- RUNAUTOMATCH

This topic describes general information for executing a job. Details for each job type are described in separate topics for individual jobs.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

This table summarizes the request parameters that are generic to all jobs. The following tables describe parameters specific to individual rules.

Table 13-5 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None

Example URL

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs`

Response


Supported Media Types: `application/json`

Parameters

This table summarizes the response parameters that are generic to all jobs. The following tables describe parameters specific to individual rules.

Table 13-6 Parameters

Name	Description
<code>jobName</code>	The name of the job, such as Create Reconciliations
<code>period</code>	The name of period, such as July2016

 **Note:**

For `monitor_reconciliations`, the parameter is `periodName`.

`filter`

The name of filter, such as `MyFilter`

 **Note:**

For `monitor_reconciliations`, the parameter is `filterName`.

Change Period Status (Reconciliation Compliance)

Changes the status of a period (open, closed, pending, locked) and returns the success or failure status.

When a period's status is changed to Open, the returned job corresponds to the opening of reconciliations for the specified period. The job's success or failure does not impact the period's status because this change is made immediately. Even if there are failures while reopening reconciliations, the period status still remains Open.

If a period's status is set to Closed or Locked, no job is returned.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-7 SET PERIOD STATUS

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, SET_PERIOD_STATUS	Yes
period	The name of the period, such as July2016	Yes
status	Status to be changed; supported values: pending, open, closed, locked	Yes

Example of request body

```
{
  "jobName" : "SET_PERIOD_STATUS",
  "parameters": {
    "period": "July2016",
    "status": "closed"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-8 Parameters

Name	Description
details	In case of errors, details are published with the error string

Table 13-8 (Cont.) Parameters

Name	Description
status	See Migration Status Codes . When the period's status is changed to Open, the status of the job that opens reconciliations for the specified period is sent as additional information. Use this additional information to check the status of the open reconciliations job.
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
jobs/2141",
      "action": "GET"
    }
  ]
}
```

Create Reconciliation (Reconciliation Compliance)

Copies all selected profiles to a period and returns success or failure status.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job. For additional parameters that are common to all jobs, see [Execute a Job](#).

Table 13-9 RULES

Name	Description	Required	Default
api_version	Version of the API you are working with, such as v1.	Yes	None
jobName	The name of a job, CREATE_RECONCILIATIONS.	Yes	None
period	The name of the period, such as July2016	Yes	None
filter	The name of filter, such as MyFilter	No	None

Example of request body

```
{
  "jobName" : "CREATE_RECONCILIATIONS",
  "parameters": {
    "period": "July2016",
    "filter": "MyFilter"
  }
}
```

Response

Parameters:

Table 13-10 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-"
    }
  ]
}
```

```

<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
jobs/2141",
  "action": "GET"
}
]
}

```

Import Pre-Mapped Balances (Reconciliation Compliance)

Imports pre-mapped balances, and returns the success or failure status.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-11 IMPORT_PREMAPPED_BALANCES

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_PREMAPPED_BALANCES	Yes
period	The name of the period, such as July2016	Yes
balanceType	Supported balance types are SRC for source system balance, and SUB for subsystem balance	Yes
currencyBucket	Currency bucket such as Functional	Yes
file	Name of the import file, such as balances.csv	Yes

Example of request body

```

{
  "jobName" : "IMPORT_PREMAPPED_BALANCES",
  "parameters": {
    "period": "July2016",
    "balanceType": "SRC",
    "file": "balances.csv",

```

```

        "currencyBucket": "Functional"
    }
}

```

Response

Supported Media Types: application/json

Parameters:

Table 13-12 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```

{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/
2141",
      "action": "GET"
    }
  ]
}

```

Import Pre-Mapped Transactions (Reconciliation Compliance)

Imports pre-mapped transactions for a particular period, and returns the success or failure status.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-13 IMPORT_PREMAPPED_TRANSACTIONS

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_PREMAPPED_TRANSACTIONS	Yes
transactionType	Transaction Type is one of the following: <ul style="list-style-type: none"> • BEX for loading Balance Explanations • SRC for loading Source System Adjustments • SUB for loading Subsystem Adjustments • VEX for loading Variance Explanations 	Yes
file	The file name, such as transactions.csv	
dateFormat	Date Format, such as DD/MM/YYYY, MMM d, yyyy, or All	Yes

Example of request body

```
{
  "jobName" : "IMPORT_PREMAPPED_TRANSACTIONS",
  "parameters": {
    "period": "July2016",
    "transactionType": "SRC",
    "file": "transactions.csv",
    "dateFormat": "MMM d, yyyy"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-14 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body**TESTINGINGING:**

```
{ "type": "ARCS", "status": -1, "details": "In Process", "links": [ { "rel":
"self", "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/2141",
"action": "GET" } ], "error": null, "items": null, "link": null }
```

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/
2141",
      "action": "GET"
    }
  ]
}
```

Import Balances (Reconciliation Compliance)

Imports balances data using Data Management from a previously created Data Load definition, and returns success or failure status.

REST Resource

```
POST /armARCS/rest/{api_version}/jobs/
```

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 13-15 IMPORT_BALANCES

Name	Description	Path	Required
api_version	Version of the API you are working with, such as v1	Yes	Yes
jobName	The name of the job, IMPORT_BALANCES	No	Yes
period	The name of the period, such as April 2016	No	Yes
dl_Definition	The name of a previously saved data load using the format DL_name .	No	Yes

Example of request body

```
{
  "jobName" : "IMPORT_BALANCES",
  "parameters": {
    "period":"April 2016",
    "dl_Definition":"DL_test"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-16 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type (self, Job Status). if set to Job Status, you can use the href to get the status of the operation
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/2141
      "action": "GET"
    }
  ]
}
```

Import Profiles (Reconciliation Compliance)

Imports profiles for a particular period, and returns the success or failure status.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-17 IMPORT_PROFILES

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_PROFILES	Yes
importType	The import type; supported values are Replace and ReplaceAll	Yes
period	The period for which to import, such as July2016	

Table 13-17 (Cont.) IMPORT_PROFILES

Name	Description	Required
profileType	The profile type; supported values are Profiles and Children	Yes
fileLocation	The file name, such as profiles.csv	Yes
dateFormat	Date Format, such as DD/MM/YYYY, MMM d, yyyy or All	Yes

Example of request body

```
{
  "jobName" : "IMPORT_PROFILES",
  "parameters": {
    "period": "July2016",
    "importType": "Replace",
    "fileLocation": "profiles.csv",
    "dateFormat": "MMM d, yyyy"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-18 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
```

```

    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/
2141",
      "action": "GET"
    }
  ]
}

```

Import Rates (Reconciliation Compliance)

Imports rates for a particular period and rate type, and returns the success or failure status.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-19 IMPORT_RATES

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_RATES	Yes
period	The name of the period, such as July2016	Yes
rateType	The rate type, such as Accounting	Yes
file	Name of the import file, such as rates.csv	Yes
importType	Supported import types are Replace and ReplaceAll	Yes

Example of request body

```

{
  "jobName" : "IMPORT_RATES",
  "parameters": {
    "period": "July2016",

```

```

        "rateType": "Accounting",
        "file": "rates.csv",
        "importType": "ReplaceAll"
    }
}

```

Response

Supported Media Types: application/json

Parameters:

Table 13-20 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```

{
  "type": "ARCS",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
jobs/2141",
      "action": "GET"
    }
  ]
}

```

Import Pre-Mapped Transactions (Transaction Matching)

Imports a file of pre-mapped transactions into Transaction Matching, and returns the success or failure status.

REST Resource

POST /arm/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-21 IMPORTTMPREMAPPEDTRANSACTIONS

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_PREMAPPED_TRANSACTIONS	Yes
dataSource	Text ID of the data source where the transaction will be imported to	
file	The file name, such as transactions.csv	
reconciliationType	Text ID of the reconciliation type where the transaction file will be imported to	Yes
dateFormat	Date Format is a parameter that includes the format of the date fields in the transactions import file. The default is DD/MM/YYYY. Other supported date formats are MM/dd/yyyy, dd/MM/yyyy, MM-dd-yyyy, d-M-yyyy, and MMM d.yyyy.	Yes

Example of request body

```
{
  "jobName" : "importtmpremappedtransactions",
  "parameters": {
    "dataSource": "CLEARING",
    "reconciliationType": "CLEARING",
    "file": "clearingTransaction.csv",
    "dateFormat": "MM-DD-YYYY"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-22 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "TM",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
jobs/2141",
      "action": "GET"
    }
  ]
}
```

Monitor Reconciliations (Reconciliation Compliance)

Returns the list of reconciliations for a given period name and filter name.

 **Note:**

- If the Reconciliation status for all the Reconciliations in the given Period Name and Filter Name were closed, then the output status would be '0'.
- If the Reconciliations status for any one of the Reconciliations in the given Period Name and Filter Name is open, then the output status would be '-1'.

REST Resource

POST /armARCS/rest/{api_version}/jobs/

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 13-23 MONITOR_RECONCILIATIONS

Name	Description	Path	Required
api_version	Version of the API you are working with, such as v1	Yes	Yes
jobName	The name of the job, MONITOR_RECONCILIATIONS	No	Yes
periodName	The name of the period, such as September 2017	No	Yes
filterName	The name of the filter. For example, Recon status filter.	No	Yes

Example of request body

```
{
  "parameters":
  {"periodName":"September 2017","filterName":"DemoFilter"},
  "jobName":"MONITOR_RECONCILIATIONS"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-24 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type

Table 13-24 (Cont.) Parameters

Name	Description
rel	Relationship type (self, Job Status). if set to Job Status, you can use the href to get the status of the operation
data	Parameters as key value pairs passed in the request

Examples of Response Body

The following is an example of the response body in JSON format when all reconciliations are closed:

```
{
  "error":null,
  "details":"Account ID : 100-1210, Name : Accounts Receivable, Status :
  Closed",
  "items":null,
  "status":0,
  "link":null,
  "links":[{"action":"GET","rel":"self","href":"https://<SERVICE_NAME>-
  <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
  jobs/monitorReconciliations/September%202017/DemoFilter"}],
  "type":"ARCS"
}
```

The following is an example of the response body in JSON format when any reconciliation status is open:

```
{
  "error":null,
  "details":"Account ID : 100-1210, Name : Accounts Receivable, Status :
  Open",
  "items":null,
  "status":-1,
  "link":null,
  "links":[{"action":"GET","rel":"self","href":"https://<SERVICE_NAME>-
  <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
  jobs/monitorReconciliations/September%202017/DemoFilter"}],
  "type":"ARCS"
}
```

Import Reconciliation Attributes (Reconciliation Compliance)

Performs flat file loads of attribute values into existing reconciliations.

REST Resource

POST /armARCS/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-25 Import Reconciliation Attributes

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, IMPORT_RECONCILIATION_ATTRIBUTES	Yes
fileName	The name of the upload import file	Yes
period	The name of the period, such as July2020	Yes
rules	The rules to run upon completion. Allowed values are: <ul style="list-style-type: none"> • ALL • SET_ATTR_VAL(Set Attribute Value) • CRT_ALT (Create Alert) • AUTO_APP(Auto Approve Reconciliation) • AUTO_SUB(Auto Submit Reconciliation) You can send multiple values separated by comma. Default is None.	No
reopen	Indicates whether to reopen changed reconciliations upon completion. Values is either true or false (default).	No
dateFormat	List of valid date formats, such as DD/MM/YYYY. You can send multiple values separated by semi-colon.	No

Here are some examples of the request body:

Example 1

```
{
  "jobName" : "IMPORT_RECONCILIATION_ATTRIBUTES",
  "parameters":
  {
    "fileName":"import_recon.csv",
    "period":"January 2010",
```

```

        "rules": "AUTO_APP,AUTO_SUB",
        "reopen": "true",
        "dateformat": "MM-dd-yyyy;MMM d, yyyy"
    }
}

```

Example 2

```

{
  "jobName" : "IMPORT_RECONCILIATION_ATTRIBUTES",
  "parameters":
  {
    "fileName": "Reconciliations.csv",
    "period": "June 2019",
    "rules": "AUTO_APP,AUTO_SUB",
    "reopen": "true"
  }
}

```

Example 3

```

{
  "jobName" : "IMPORT_RECONCILIATION_ATTRIBUTES",
  "parameters":
  {
    "fileName": "import_recon.csv",
    "period": "January 2010",
    "rules": "ALL",
    "reopen": "true"
  }
}

```

Example 4

```

{
  "jobName" : "IMPORT_RECONCILIATION_ATTRIBUTES",
  "parameters":
  {
    "fileName": "Reconciliations.csv",
    "period": "June 2019"
  }
}

```

Response

Supported Media Types: application/json

Parameters:

Table 13-26 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	<ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "ARCS",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/
100000001155018",
      "action": "GET"
    }
  ],
  "details": "In Process",
  "status": -1
}
```

Run Auto Match (Transaction Matching)

Runs the auto match process in Transaction Matching.

REST Resource

POST /arm/rest/{api_version}/jobs

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-27 runautomatch

Name	Description	Required
api_version	Version of the API you are working with, such as v1	Yes
jobName	The name of a job, runautomatch	Yes
ReconTypeId	The Text ID of the Reconciliation type to be auto matched	Yes

Example of request body

```
{"jobName":"runautomatch",
"parameters":{"reconTypeId":"INTERCO"}}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-28 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "TM",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/v1/jobs/
2141",
      "action": "GET"
    }
  ],
  "error": null,
  "items": null,
}
```

```

    "link": null
  }

```

Purge Transactions (Transaction Matching)

Removes matched transactions for Account Reconciliation using the provided parameters.

You can specify `filterOperator` and `filterValue` to further filter the matched transactions. Otherwise, all matched transactions older than or equal to the age from all accounts for the specified `matchType` are purged.

REST Resource

```
POST /arm/rest/{api_version}/jobs
```

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Request

Supported Media Types: `application/json`

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-29 Parameters

Name	Description	Data Type	Required	Values
<code>api_version</code>	Version of the API you are working with, such as v1	String	Yes	v1
<code>jobName</code>	The name of a job	String	Yes	<code>purgetransactions</code>
<code>ReconTypeId</code>	The identifier (TextID) of the match type from which matched transactions should be deleted	String	Yes	Example: <code>CLEARING</code>
<code>matchedStatus</code>	The status of the transactions to be deleted. The only value currently supported is <code>matched</code> .	String	No	Default value: <code>matched</code>
<code>age</code>	Identifies the number of days since the transaction was matched. Matched transactions older than or equal to this value will be deleted.	Integer	Yes	Example: 180

Table 13-29 (Cont.) Parameters

Name	Description	Data Type	Required	Values
filterOperator	<p>Optionally, use one of the following filter conditions to identify the accounts containing matched transactions for deletion. This value is combined with the filterValue to identify the accounts from which matched transactions should be purged:</p> <ul style="list-style-type: none"> • EQUALS • NOT_EQUALS • STARTS_WITH • ENDS_WITH • CONTAINS • NOT_CONTAINS 	String	No	<p>Examples:</p> <p>EQUALS</p> <p>Starts_With</p>
filterValue	<p>Optionally use one or more filter values to identify the transactions to purge. Use a space-separated list to specify multiple values as shown in the examples. If multiple values are specified, transactions from accounts matching any filter operator and filter value combination are selected for purging.</p> <p>EQUALS and NOT_EQUALS support multiple values.</p> <p>STARTS_WITH, ENDS_WITH, CONTAINS, and NOT_CONTAINS only support a single value.</p>	String	No	<p>Examples:</p> <p>["101-1234", "102-1234"]</p> <p>["102"]</p>
logFileName	<p>Optionally, enter the name of a log file to record information about the API activity.</p>	String	No	<p>If a file name is not specified, a log file named PurgeTransactions_<i>Job_ID</i> is automatically generated. The file is located in the Outbox.</p>

Example of request body

Example of purging matched transactions 120 days or older for CLEARING match type and accounts equal to 101-1234 and 102-1234:

```
{
  "jobName" : "purgetransactions",
  "parameters":
  {
    "reconTypeId" : "CLEARING",
    "age" : 120,
    "filterOperator":"EQUALS",
    "filterValue" : ["101-1234","102-1234"],
    "logFileName" : "purge-clearing"
  }
}
```

Example of purging matched transactions 120 days or older for CLEARING match type and accounts start with 101:

```
{
  "jobName" : "purgetransactions",
  "parameters":
  {
    "reconTypeId" : "CLEARING",
    "age" : 120,
    "filterOperator":"STARTS_WITH",
    "filterValue" : ["101"],
    "logFileName" : "purge-clearing"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-30 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "type": "TM",
  "status": -1,
  "details": "In Process",
  "links": [
    {
      "rel":
"self",
      "href": "https://
<SERVICE_NAME><TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/
rest/v1/jobs/2141",
      "action": "GET"
    }
  ],
  "error": null,
  "items": null,
  "link": null
}
```

Retrieve Job Status for Account Reconciliation

Returns the status of a job for Account Reconciliation, indicating if the job is in process, or if it is successfully executed or completed with errors.

REST Resource

GET /armARCS/rest/{api_version}/jobs/{job_id}

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 13-31 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
jobIdentifier	The ID of the job	Path	Yes	None

Response

Parameters

The following table summarizes the response parameters.

Table 13-32 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Supported Media Types: application/json

Example of Response Body

```
{
  "type": "ARCS",
  "status": 0,
  "details": "Total to copy : 3\nSuccessfully copied : 2\nUnsuccessfully
copied : 1\n",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/jobs/91",
      "action": "GET"
    }
  ],
  "error": null,
  "items": null,
  "link": null
}
```

Export Application Properties

Exports Account Reconciliation application settings (related to Redwood Experience, theme, email notification, and business process name), background image, and logo image to a JSON file so that you can import them into the same or another environment. This command is useful when you import an application from prod to test environments. If your application settings are different in prod and test environments, you can export them from the test environment before importing the application from the prod environment, and then import the settings in to the test environment to maintain the original settings.

You can download the export file using the Download REST API. This is a synchronous API.

REST Resource

POST /arm/rest/fcmapi/{api_version}/rc/export/applicationproperties

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-33 EXPORT APPLICATION PROPERTIES

Name	Description	Required	Default
fileName	The name of the JSON file that stores the exported property values. Use the Upload API to upload the file to the target environment and then restore these settings in the target environment, as described in Import Application Properties .	Yes	None
properties	The comma-separated list of properties to be exported. Valid values include the following: <ul style="list-style-type: none"> • Theme exports the display theme used in the environment • EmailNotification exports the email notification settings defined in the environment • DisplayBusinessProcessesName exports whether to display the business process name on the page in the environment • RedwoodExperience exports the Redwood Experience setting of the environment • BackgroundImage exports the background image used in the environment • LogoImage exports the logo image used in the environment 	No	All valid properties are exported.

Examples of request body

Example 1: Exporting all exportable application properties

```
{
  "fileName": "ApplicationProperties.json"
}
```

Example 2: Exporting specific application properties

```
{
  "fileName": "ApplicationProperties.json",
  "properties":["Theme", "EmailNotification", "DisplayBusinessProcessName",
"RedwoodExperience"]
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-34 Parameters

Name	Description
details	In case of errors, details are published with the error string.
status	Status of the job: <ul style="list-style-type: none"> • -1 = In Progress • 0 = Success • 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. It can be <code>self</code> or <code>export-content</code> . If the export succeeds, you can use the <code>href</code> to download the exported file.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details": "Application properties exported successfully",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
export/applicationproperties",
      "action": "POST"
    },
    {
      "rel": "export-content",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/"
```

```

ApplicationProperties.json/contents",
    "action": "GET"
  }
],
"status": 0,
"type": "RC",
"link": {},
"error": null,
"items": []
}

```

Import Application Properties

Imports Account Reconciliation application settings (related to Redwood Experience, theme, email notification, and business process name), background image, and logo image from an export file into an Account Reconciliation environment. The export file is available for import after the file is uploaded using the Upload REST API.

This a synchronous API.

REST Resource

```

POST /arm/rest/fcmapi/{api_version}/rc/import/
applicationproperties

```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-35 IMPORT APPLICATION PROPERTIES

Name	Description	Required	Default
api_version	The REST API version for the API. This release is v1.	Yes	v1
fileName	The name of the JSON file that contains exported property values from another environment. This file, exported from another environment as described in Export Application Properties , must be available in the environment where you are restoring application settings.	Yes	None

Example of request body

```
{
  "fileName": "applicationProperties.json"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-36 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	Status of the job: <ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. Possible values: self.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details": "Application properties imported successfully",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
import/applicationproperties",
      "action": "POST"
    }
  ],
  "status": 0,
  "type": "RC",
  "link": {},
  "error": null,
  "items": []
}
```

Export Background Image

Exports the background image used in an Account Reconciliation environment to a JPG file so that you can import it into another environment. You can download the image file using the Download REST API.

This is a synchronous API.

REST Resource

POST /arm/rest/fcmap/{api_version}/rc/export/backgroundImage

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-37 EXPORT BACKGROUND IMAGE

Name	Description	Required	Default
api_version	The current REST API version for the API. For example, v1 for this API.	Yes	v1
fileName	The name for the background image file in JPG, JPEG, GIF, or PNG format. Use the Upload API to upload the background image file to the target environment and then import it into the target environment, as described in Import Background Image .	Yes	None

Example of request body

```
{
  "fileName": "backgroundImage.jpg"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-38 Parameters

Name	Description
details	In case of errors, details are published with the error string

Table 13-38 (Cont.) Parameters

Name	Description
status	Status of the job: <ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. It can be <code>self</code> or <code>export-content</code> . If the export succeeds, you can use the <code>href</code> to download the exported file.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "details": "Background Image exported successfully",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
export/backgroundImage",
      "action": "POST"
    },
    {
      "rel": "export-content",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/bgImage.jpg/contents",
      "action": "GET"
    }
  ],
  "status": 0,
  "type": "RC",
  "link": {},
  "error": null,
  "items": []
}
```

Import Background Image

Imports the background image from an export file into an Account Reconciliation environment and then sets it as the current background image. The image will be available for import after you upload the file using the Upload REST API.

This is a synchronous API.

REST Resource

POST /arm/rest/fcmapi/{api_version}/rc/import/backgroundImage

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-39 IMPORT BACKGROUND IMAGE

Name	Description	Required	Default
api_version	The current REST API version for the API. For example, v1 for this API.	Yes	v1
fileName	The name of the background image file that was exported from another environment. Supported formats include JPG, JPEG, GIF, and PNG.	Yes	None

Example of request body

```
{
  "fileName": "backgroundImage.jpg"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-40 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	Status of the job: <ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. Possible values: self.

Example of Response Body

The following is an example of the response body in JSON format.

```

{
  "details": "Background image imported successfully.",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
import/backgroundImage",
      "action": "POST"
    }
  ],
  "status": 0,
  "type": "RC",
  "link": {},
  "error": null,
  "items": []
}

```

Export Logo Image

Exports the corporate logo used in an Account Reconciliation business process to a JPG file so that you can import it into another environment. The exported file can be download using the Download REST API.

This is an asynchronous API.

REST Resource

POST /arm/rest/fcmapi/{api_version}/rc/export/logo

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-41 EXPORT LOGO IMAGE

Name	Description	Required	Default
api_version	The current REST API version for the API. For example, v1 for this API.	Yes	v1

Table 13-41 (Cont.) EXPORT LOGO IMAGE

Name	Description	Required	Default
fileName	The name of the logo image file in JPG, JPEG, GIF, or PNG format. Use the Upload API to upload the background image file to the target environment and then import it into the target environment, as described in Import Logo Image .	Yes	None

Example of request body

```
{
  "fileName": "logo.jpg"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-42 Parameters

Name	Description
details	In case of errors, details are published with the error string.
status	Status of the job: <ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. It can be <code>self</code> or <code>export-content</code> . If the export succeeds, you can use the <code>href</code> to get the status of the import operation.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/
fcmapi/v1/RC/export/logo",
```

```

        "action": "POST"
      },
      {
        "rel": "export-content",
        "href": "<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/
applicationsnapshots/logo.jpg/contents",
        "action": "GET"
      }
    ],
    "details": "Logo image exported successfully.",
    "type": "RC",
    "status": 0,
    "link": {},
    "error": null,
    "items": []
  }
}

```

Import Logo Image

Imports the corporate logo used in an Account Reconciliation environment from an export file into another environment. The logo is available for import after you upload the image file using the Upload REST API.

This is an asynchronous API.

REST Resource

POST /arm/rest/fcmapi/{api_version}/rc/import/logo

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 13-43 IMPORT LOG IMAGE

Name	Description	Required	Default
api_version	The current REST API version for the API. For example, v1 for this API.	Yes	v1
fileName	The name of the logo image file that contains the image to be imported. Supported formats include JPG, JPEG, GIF, and PNG.	Yes	None

Example of request body

```
{
  "fileName": "logo.jpg"
}
```

Response

Supported Media Types: application/json

Parameters:

Table 13-44 Parameters

Name	Description
details	In case of errors, details are published with the error string.
status	Status of the job: <ul style="list-style-type: none"> -1 = In Progress 0 = Success 1 = Fail
links	Detailed information about the link
href	Links to API call or status API
action	The HTTP call type
rel	Relationship type. Possible values: self.

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "links": [
    {
      "rel": "self",
      "href": "<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/
fcmapi/v1/RC/import/logo",
      "action": "POST"
    }
  ],
  "details": "Logo image imported successfully.",
  "type": "RC",
  "status": 0,
  "link": {},
  "error": null,
  "items": []
}
```

Working with Connections in Account Reconciliation

Use these REST APIs to work with connections.

With multiple environments, using REST APIs saves you time and effort by automating the process of logging in and configuring connections. For information about accessing environments, see [Accessing EPM Cloud](#).

Table 13-45 Working with Connections in Account Reconciliation

Task	Request	REST Resource
Create a Connection	POST	/arm/rest/fcmapi/{api_version}/{module}/connections
View All Connections	GET	/arm/rest/fcmapi/{api_version}/{module}/connections
Update a Connection	PUT	/arm/rest/fcmapi/{api_version}/{module}/connections/{id}
Delete a Connection	DELETE	/arm/rest/fcmapi/{api_version}/{module}/connections/{id}

Create a Connection

Use this REST API to create a connection that will be saved in an application.

REST Resource

POST /arm/rest/fcmapi/{api_version}/{module}/connections

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request parameters specific to this job.

Table 13-46 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None
module	The name of the module for which to create a connection. For Account Reconciliation, set this value to RC.	Path	Yes	None
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com	Payload	Yes	None
username	A username with the Service Administrator predefined role	Payload	Yes	None

Table 13-46 (Cont.) Parameters

Name	Description	Type	Required	Default
password	The encrypted password for the user For security reasons, only an encrypted password is allowed. Use the EPM Automate encrypt command to generate the encrypted password. See encrypt .	Payload	Yes	None
type	The type of connection. Set the value to ENTERPRISE_JOURNALS.	Payload	Yes	None

Example URL

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/
connections
```

Example of Request Body

```
{
  "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com",
  "username": "<NEW_USERNAME>",
  "password": "<NEW_PASSWORD>",
  "type" : "ENTERPRISE_JOURNALS"
}
```

Response

Supported Media Type: application/json

Table 13-47 Parameters

Parameters	Description
details	In case of errors, details are published with the error string.

Example of Response Body

```
{
  "details": "Connection created successfully."
}
```

View All Connections

Use this REST API to view details for all of the connections saved in an application.

Required Roles

Service Administrator

REST Resource

```
GET /arm/rest/fcmapi/{api_version}/{module}/connections
```

Request

Parameters:

The following table summarizes the client request.

Table 13-48 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None
module	The name of the module for which to view connections. For Account Reconciliation, set this value to RC.	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/connections
```

Response

Supported Media Types: application/json

The following table summarizes the parameters.

Table 13-49 Parameters

Parameters	Description
items	Collection of information about the resource
id	Unique identifier for the connection, such as 1c89922d-92ba-46c1-850f-e2a8a416ddf2
type	The type of connection
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com
links	Detailed information about the link
details	In case of errors, details are published with the error string

Example Response

```
{
  "links": [
    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/connections",
      "action": "GET"
    }
  ]
}
```



```

    ],
    "details": null,
    "items": [
      {
        "username": "ats_admin2",
        "password": null,
        "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com",
        "id": 100000000558005,
        "type": "ENTERPRISE_JOURNALS"
      }
    ]
  }
}

```

Update a Connection

Use this REST API to update a specific connection that is saved in an application.

REST Resource

```
PUT /arm/rest/fcmapi/{api_version}/{module}/connections/{id}
```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request.

Table 13-50 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None
module	The name of the module for which to update a connection. For Account Reconciliation, set this parameter to RC.	Path	Yes	None
id	The identifier of the connection that must be updated	Path	Yes	None
url	The URL of the connection, such as https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com	Payload	Yes	None
username	A username with Service Administrator predefined role	Payload	Yes	None
password	The encrypted password for the user For security reasons, only an encrypted password is allowed. Use the EPM Automate encrypt command to generate the encrypted password. See encrypt .	Payload	Yes	None

Table 13-50 (Cont.) Parameters

Name	Description	Type	Required	Default
type	The type of connection. Set the value to ENTERPRISE_JOURNALS.	Payload	Yes	None

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/connections/100000000658005
```

Example of Request Body

```
{
  "url": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com",
  "username": "<NEW_USERNAME>",
  "password": "<NEW_PASSWORD>"
}
```

Response

Supported Media Type: application/json

Table 13-51 Parameters

Parameters	Description
details	In case of errors, details are published with the error string.

Example of Response Body**Example 1:**

```
{
  "details": "Connection updated successfully."
}
```

Example 2:

```
{
  "details": "Invalid parameters. Test Connection is failed"
}
```

Delete a Connection

Use this REST API to delete a specific connection that is saved in an application.

Required Roles

Service Administrator

REST Resource

```
DELETE /arm/rest/fcmapi/{api_version}/{module}/connections/{id}
```

Request

Parameters:

The following table summarizes the client request.

Table 13-52 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None
module	The name of the module for which to delete a connection. For Account Reconciliation, set this parameter to RC.	Path	Yes	None
id	The identifier of the connection that must be deleted	Path	Yes	None

Example URL

```
https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/arm/rest/fcmapi/v1/rc/  
connections/100000000552006
```

Response

Supported Media Type: application/json

Table 13-53 Parameters

Parameters	Description
details	In case of errors, details are published with the error string.

Example of Response Body

```
{  
  "details": "Connection deleted successfully."  
}
```

Set Application Access Level

Sets the access level for an Account Reconciliation application. You can specify that the application can be accessed only by Service Administrators or by all users.

REST Resource

```
POST /armARCS/rest/{api_version}/appaccess
```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters:

The following table summarizes the client request parameters.

Table 13-54 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None
access	Specify the access level for users. Valid values include the following: <ul style="list-style-type: none"> ALL_USERS - all users can access the application ADMINISTRATORS - only Service Administrators can access the application 	Payload	Yes	None

Examples of Request Body

Example 1

```
{
  "access": "ALL_USERS"
}
```

Example 2

```
{
  "access": "ADMINISTRATORS"
}
```

Response

Supported Media Type: application/json

Table 13-55 Parameters

Parameters	Description
details	In case of errors, details are published with the error string.
status	See Migration Status Codes
links	Detailed information about the link

Example of Response Body

```
{
  "status": 0,
  "details": "Application access updated successfully",
  "links": [
```

```

    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
appaccess",
      "action": "GET",
      "data": null
    }
  ]
}

```

Retrieve Application Access Level

Returns the access level for an Account Reconciliation application, indicating if all users can access the application or only Service Administrators can access the application.

REST Resource

GET /armARCS/rest/{api_version}/appaccess

Required Roles

Service Administrator

Request

Parameters:

The following table summarizes the client request parameters.

Table 13-56 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with (must be v1)	Path	Yes	None

Example URL

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/appaccess

Response

Supported Media Type: application/json

Table 13-57 Parameters

Parameters	Description
links	Detailed information about the link
access	Indicates the access level for users. <ul style="list-style-type: none"> ALL_USERS - all users can access the application ADMINISTRATORS - only Service Administrators can access the application

Example of Response Body

```
{
  "items": [
    {
      "access": "ALL_USERS"
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "http://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/armARCS/rest/v1/
appaccess",
      "action": "GET",
      "data": null
    }
  ]
}
```

Financial Consolidation and Close REST APIs

Use the Financial Consolidation and Close REST APIs to get the REST API version, retrieve journals and journal details, submit, approve, post, unpost, and reject journals, and update journal periods. You can also import supplementation data, copy data, clear data, and deploy form templates.

Getting API Versions for Financial Consolidation and Close APIs

You can get information on REST API versions using REST resources. See [Getting API Versions for Planning](#). Financial Consolidation and Close APIs use the same version numbers as Planning.

Table 14-1 Task Manager, Supplemental Data Manager, and Enterprise Journal APIs

API Name	Versions
Import Supplemental Collection Data for Financial Consolidation and Close	v3
Deploy Form Templates	v3
Deploy Task Manager Templates	v1
Update Task Status for Event Monitoring	v1
Execute an Enterprise Journals Job	v1
Monitor Enterprise Journals for Financial Consolidation and Close	v1
Retrieve Enterprise Journals for Financial Consolidation and Close	v1
Retrieve Enterprise Journal Content for Financial Consolidation and Close	v1
Retrieve Enterprise Journal Content by Year and Period for Financial Consolidation and Close	v1
Update Enterprise Journal Posting Status for Financial Consolidation and Close	v1

Getting API Versions for Financial Consolidation and Close

Financial Consolidation and Close uses Planning REST APIs that use the Planning REST API version. See [Getting API Versions for Planning](#).

Get Information about a Specific API Version for Financial Consolidation and Close APIs

Returns details for a specific REST API version for Financial Consolidation and Close.

REST Resource

GET /HyperionPlanning/rest/{api_version}

Required Roles

Service Administrator, Power User, User, Viewer

Request

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 14-2 Parameters

Name	Description
api_version	Version of the API you are developing with, such as V1

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 14-3 Parameters

Name	Description
version	The version, such as V1

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "version": "v1",
  "lifecycle": "active",
  "isLatest": true,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
    <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
    rest/v2"
  }, {
    "rel": "predecessor-version",
    "href": "https://<SERVICE_NAME>-
    <TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
    rest/v1"
```



```

    }]
  }

```

Import Supplemental Collection Data for Financial Consolidation and Close

Imports supplemental data to a collection for the frequency dimensions defined in the collection interval of the collection. Returns the success or failure status.

REST Resource

```

POST      /HyperionPlanning/rest/{api_version}/applications/{application}/
fcmjobs

```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

Table 14-4 IMPORT_SUPPLEMENTAL_COLLECTION_DATA

Name	Description	Type	Required	Default
api_version	Version of the API you are working with: v3	Path	Yes	None
application	The name of the application Get the application name by using the Get Applications API, for example, FCCS or TRCS. See Get Applications .	Path	Yes	None
jobName	Name of the job	String	No	None
jobType	Type of the Job. Supported value for this Job IMPORT_SUPPLEMENTAL_COLLECTION_DATA	String	Yes	None
fileName	File name to be imported, for example, dataset.csv	String	Yes	None
collection	Collection or sub-collection name	String	Yes	None
year	The year for which the collection is deployed	String	Yes	None

Table 14-4 (Cont.) IMPORT_SUPPLEMENTAL_COLLECTION_DATA

Name	Description	Type	Required	Default
period	Period name for which the collection is deployed	String	Yes	None
parameter	Runtime parameter. This is the frequency dimension used for the Collection.	String	No	None
value	Runtime parameter value. This is the member value for the dimension specified in the parameter.	String	No	None

Table 14-5 Examples of runtime parameters

Parameter	Value
Product	Oracle EPM
Consolidation	Entity Input
Movement	Actual

Examples of request body

Example 1

```
{
  "jobType" : "IMPORT_SUPPLEMENTAL_COLLECTION_DATA",
  "parameters": {
    "fileName":"import_sdm_data.csv",
    "collection":"Investment Detail
Collection",
    "year":"2019",
    "period":"Dec",
    "product":"Oracle EPM",
    "consolidation":"Entity Input",
  }
}
```

Example 2

```
{
  "jobType" : "IMPORT_SUPPLEMENTAL_COLLECTION_DATA",
  "parameters": {
    "fileName":"import_sdm_data.csv",
    "collection":"Investment Detail
Collection",
    "year":"2020",
    "period":"January",
    "category":"Oracle EPM",
    "movement":"Actual",
  }
}
```

```

    }
}

```

Example 3

```

{
  "jobType" : "IMPORT_SUPPLEMENTAL_COLLECTION_DATA",
  "parameters": {
    "fileName": "import_sdm_data.csv",
    "collection": "Investment Detail Collection",
    "year": "2019",
    "period": "December",
    "scenario": "Actual",
  }
}

```

Response

Supported Media Types: application/json

Parameters:

Table 14-6 Parameters

Name	Description
type	FCCS Application type, for example, FCCS
status	-1 - In Progress; 0 - Success; 1 - Fail
details	In case of errors, details are published with the error string
descriptiveStatus	The status of the job, such as Completed or Error
items	Collection of Notification categories
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type. Possible values: self

Example of Response Body

The following is an example of the response body in JSON format.

```

{
  "jobId": 100000000114040,
  "descriptiveStatus": "In Progress",
  "details": "In Progress",
  "status": -1,
  "items": [],
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME-
<TENANT_NAME.<SERVICE_TYPE.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/FCCS/fcmjobs/100000000114040"
    }, {"

```

```

        "action": "GET"
    }
  ],
}

```

Perform Journal Actions for Financial Consolidation and Close

Performs journal action for the specified journal. Changes the journal status to the new state specified.

Journal actions supported: SUBMIT, APPROVE, POST, UNPOST, REJECT

This API works only for Financial Consolidation and Close.

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/journals/{journalLabel}/actions

Required Roles

Role	Available Actions
Service Administrator	All
Power User	Submit/Post/Unpost: Need Write access to all the members in the journal Approve/Reject. Need Read access to all the members in the journal
User	Approve/Reject. Need Read access to all the members in the journal and Journal option must have been enabled

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 14-7 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
journalLabel	Label for the journal	Path	Yes	None

Table 14-7 (Cont.) Parameters

Name	Description	Type	Required	Default
parameters	Parameters for journals, for example: <pre> {"parameters" :{ "scenario": "Actual", "year": "FY17", "period": "Jan", "consolidation": "FCCS_Entity Input", "action": "POST" } } </pre>	JSON	Yes	None
	These parameter fields are described in rows below.			
scenario	The scenario for which you are performing the journal action	Query	Yes	None
year	The year for which you are performing the journal action	Query	Yes	None
period	The period for which you are performing the journal action	Query	Yes	None
consolidation	The consolidation for which you are performing the journal action	Query	No	Entity Input
action	The journal action. Supported valid Action values: SUBMIT,REJECT,APPROVE,POST,UNPOST	Query	Yes	None

Response

Parameters

The following table summarizes the response parameters.

Table 14-8 Parameters

Name	Description
actionDetail	Journal action, such as Posted
actionStatus	Action status, such as 0

Supported Media Types: application/json

Sample JSON Input

```

{
  "scenario": "Actual",

```

```

    "year": "FY17",
    "period": "Jan",
    "action": "POST"
  }

```

Example of Response Body

The following shows an example of the response body.

```

{
  "actionDetail": "Posted",
  "actionStatus": 0
}

```

Perform Journal Period Updates for Financial Consolidation and Close

Performs journal period update action for the specified period.

Journal period actions supported: OPEN,CLOSE

This API works only for Financial Consolidation and Close.

REST Resource

```

POST      /HyperionPlanning/rest/{api_version}/applications/
{application}/journalPeriods/{period}/actions

```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 14-9 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
period	The period for which you are performing the journal action	Path	Yes	None

Table 14-9 (Cont.) Parameters

Name	Description	Type	Required	Default
parameters	Parameters for the journal period, for example: <pre>{ "parameters": { "scenario": "Actual", "year": "FY17", "action": "OPEN" } }</pre>	JSON	Yes	None
	These parameters are described in rows below.			
scenario	The scenario for which you are performing the journal action	JSON	Yes	None
year	The year for which you are performing the journal action	JSON	Yes	None
action	The journal period action. Supported valid Action values: OPEN, CLOSE	JSON	Yes	None

Response**Parameters**

The following table summarizes the response parameters.

Table 14-10 Parameters

Name	Description
scenario	Journal scenario, such as Actual
year	Journal year, such as FY18
period	Journal period, such as Jan
action	Journal period action, such as Open

Supported Media Types: application/json

Sample JSON Input

```
{
  "scenario": "Actual",
  "year": "FY17",
  "period": "Jan",
  "action": "OPEN"
}
```

Example of Response Body

The following shows an example of the response body.

```
{
  "actionDetail": "Open",
  "actionStatus": 0
}
```

Monitor Enterprise Journals for Financial Consolidation and Close

Returns the status of Journals for the given parameters.

 **Note:**

- If all journals for given parameters are closed, then the output status would be '0' and detail text specifies that all journals are closed.
- If any journal for the given parameters is in Open status (Pending, Open with Preparer, Open with Approver etc.), then the output status would be '-1'.
- In case of error, a positive number will be returned in status.

REST Resource

POST /rest/ej/{api_version}/jobs/

Required Roles

Service Administrator and Power User

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 14-11 Parameters

Name	Description	Path	Required
api_version	Version of the API you are working with, such as v1.	Yes	Yes
jobType	The name of the job EJ_MONITOR_JOURNALS.	No	Yes
year	The name of the year, such as 2021, 2022 etc.	No	No
period	The name of the period, such as Jan, Feb etc.	No	No

Table 14-11 (Cont.) Parameters

Name	Description	Path	Required
filterName	The name of the filter. For example, Journal status filter.	No	Yes



Note:

Either the combination of Year and Period or the Filter Name must be passed. If Year and Period can be part of filter, then Filter Name is sufficient in the parameters.

Example of request body

```
{
  "jobType": "EJ_MONITOR_JOURNALS",
  "parameters": {
    "year": "2022",
    "period": "Jan",
    "filterName": "Jan22_ERPDirectJournals"
  }
}
```

Response

Supported Media Types: application/json

Parameters:

Table 14-12 Parameters

Name	Description
details	Journal Name and Status. First 100 journals only. Shows Open journals first. In case of errors, details are published with the error string.
status	See Migration Status Codes
links	Detailed information about the link
href	Links to the API call
action	The HTTP call type
rel	Relationship type (self, Job Status). if set to Job Status, you can use the href to get the status of the operation

Examples of Response Body

The following is an example of the response body in JSON format when all journals are closed:

```
{
  "status": 0,
  "details": "All journals for the given filter are closed.",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>/HyperionPlanning/rest/ej/
{api_version}/jobs/rest/ej/v1/jobs/monitorJournals?
year=2022&period=Jan&filterName=ERP_Direct",
      "action": "GET"
    }
  ],
  "type": "EPM",
}
```

The following is an example of the response body in JSON format when any journal status is open:

```
{
  "status": -1,
  "details": "Journal_NonDirect1|2022|Jan|1000000001, Status:Open
with Preparer; \n
Journal_NonDirect1|2022|Jan|1000000002, Status:Closed",
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com
HyperionPlanning/rest/ej/{api_version}/jobs/monitorJournals?
year=2022&period=Jan&filterName=EJ_NonDirect ",
      "action": "GET"
    }
  ],
  "type": "EPM",
}
```

Retrieve Journals for Financial Consolidation and Close

Returns the list of journals for the given scenario, year, period, journal status and other specified filters.

Paging is supported if the optional offset and limit parameters are provided.

This API works only for Financial Consolidation and Close.

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/
journals?
q={"scenario":"Actual","year":"FY16","period":"Jan","consolidation":"FCCS_En
tity Input","status":
"WORKING","group":"group1","label":"J1","description":"JournalDesc","entity
":"FCCS_Total Geography"}&offset=0&limit=5
```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 14-13 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
q	Filters for retrieving the journals, for example: {"scenario":"Actual","year":"FY16","period":"Jan","consolidation":"FCCS_Entity Input","status":"WORKING","group":"group1","label":"J1","description":"JournalDesc","entity":"FCCS_Total Geography"}	Query	No	5 journals are returned.
scenario	Possible values are described in the following rows. Scenario for the journal, for example: "scenario":"Actual"	Query	Yes	None
year	Year for the journal, for example: "year":"FY16"	Query	Yes	None
period	Period for the journal, for example: "period":"Jan"	Query	Yes	None
consolidation	Consolidation for the journal, for example: "consolidation":"FCCS_Entity Input"	Query	No	Entity Input

Table 14-13 (Cont.) Parameters

Name	Description	Type	Required	Default
status	Status for the journal, for example: "status": "WORKING" Valid values are "WORKING", "SUBMITTED", "POSTED", "APPROVED"	Query	Yes	None
group	Group for the journal, for example: "group": "group1"	Query	No	None
label	Label for the journal, for example: "label": "j1"	Query	No	None
description	Description for the journal, for example: "description": "adjustment for salary"	Query	No	None
entity	Entity for the journal, for example: "entity": "FCCS_Total Geography"	Query	No	None
offset	Used for pagination of the records. Indicates the actual index from which the records are returned. It is 0 based.	Query	No	0
limit	Used for pagination of the records. Controls how many items to return. Defaults to 5 if not specified.	Query	No	5

Response

The following table summarizes the response parameters.

Table 14-14 Parameters

Name	Description
totalResults	Total number of journals matching the filter criteria
hasMore	True/False, if there are more pages of records
count	Number of journals in this page
limit	Current page size
offset	Current page number

Table 14-14 (Cont.) Parameters

Name	Description
items	<p>List of journals, followed by attributes of journals, such as below:</p> <pre data-bbox="672 436 1354 938"> [[{ "scenario": "Actual", "currency": "Entity Currency", "createdOn": "2018-07-30 06:22:47.516", "modifiedBy": "epm_default_cloud_admin", "journalType": "Regular", "createdBy": "epm_default_cloud_admin", "balanceType": "Balanced", "postedBy": null, "year": "FY17", "description": "JournalDesc1", "group": "grpl", "status": "Working", "label": "J4", "period": "Jan"]]] </pre>

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body.

```

{
  "totalResults": 10
  "hasMore": false,
  "count": 5,
  "limit": 5,
  "offset": 0,
  "items": [{
    "scenario": "Actual",
    "createdOn": "2018-07-30 06:22:47.516",
    "modifiedBy": "epm_default_cloud_admin",
    "journalType": "Regular",
    "createdBy": "epm_default_cloud_admin",
    "balanceType": "Balanced",
    "postedBy": null,
    "year": "FY17",
    "description": "JournalDesc1",
    "group": "grpl",
    "status": "Working",
    "label": "J4",
    "period": "Jan",
    "journalUrl": {,
    "rel": "Journal Item",
    "href": "https://<SERVICE_NAME-

```

```

<TENANT_NAME.<SERVICE_TYPE.<dcX>.oraclecloud.com/HyperionPlanning/
faces/LogOn?SO_jumpToEfsStructureHome=Y&SO_efsJumpToCardId=EPM_CA_6",",
  "data": null,
  "action": "GET",
  }
},
{
  "scenario": "Actual",
  "currency": "Entity Currency",
  "createdOn": "2018-07-26 10:21:35.634",
  "modifiedBy": "epm_default_cloud_admin",
  "journalType": "Regular",
  "createdBy": "epm_default_cloud_admin",
  "balanceType": "Balanced",
  "postedBy": null,
  "year": "FY17",
  "description": "JournalDesc1",
  "group": "grp1",
  "status": "Working",
  "label": "J2",
  "period": "Jan",
  "journalUrl": {,
  "rel": "Journal Item",
  "href": "https://<SERVICE_NAME-
<TENANT_NAME.<SERVICE_TYPE.<dcX>.oraclecloud.com/HyperionPlanning/
faces/LogOn?SO_jumpToEfsStructureHome=Y&SO_efsJumpToCardId=EPM_CA_6",",
  "data": null,
  "action": "GET",
  },
  }
],
"links": [
{
  "rel": "Get Journals",
  "href": "https://<SERVICE_NAME-
<TENANT_NAME.<SERVICE_TYPE.<dcX>.oraclecloud.com/HyperionPlanning/
rest/v3/applications/BotApp/journals",
  "action": "GET
  }
],
}

```

Retrieve Journal Details for Financial Consolidation and Close

Returns the journal details for the given scenario, year, period, consolidation, and the journal name.

This API works only for Financial Consolidation and Close.

REST Resource

```
GET /HyperionPlanning/rest/{api_version}/applications/{application}/
journals/{journal label}?
q={"scenario":"Actual","year":"FY16","period":"Jan","consolidation":"Entity
Input","lineItems":"true"}
```

Required Roles

Service Administrator

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 14-15 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	The name of the application	Path	Yes	None
journal label	The label of the journal for which to retrieve journal details	Path	Yes	None
q	Filters to retrieve the journal, for example: q={"scenario":"Actual","year":"FY16","period":"Jan","consolidation":"Entity Input"}	Query	Yes	None
scenario	Scenario for the journal, for example: "scenario":"Actual"	Query	Yes	None
year	Year for the journal, for example: "Year":"FY16"	Query	Yes	None
period	Period for the journal, for example: "period":"Jan"	Query	Yes	None
consolidation	Consolidation for the journal, for example: "consolidation":"Entity Input"	Query	No	Entity Input

Table 14-15 (Cont.) Parameters

Name	Description	Type	Required	Default
lineItems	Line items for the journal. Valid values are true or false.	Query	No	True

Response

Parameters

The following table summarizes the response parameters.

Table 14-16 Parameters

Name	Description
	<p>First-level attributes related to the journal, such as:</p> <pre>[{ "scenario": "Actual", "createdOn": "2018-07-30 06:22:47.516", "modifiedBy": "epm_default_cloud_admin", "journalType": "Regular", "createdBy": "epm_default_cloud_admin", "balanceType": "Balanced", "postedBy": null, "year": "FY17", "description": "JournalDesc1", "group": "grp1", "status": "Working", "label": "J4", "currency": "Entity Currency", "period": "Jan"}]</pre>
lineItems	<p>List of line items for the journal, followed by line items attributes, such as:</p> <pre>[{ "account": "Balance", "amount": "200", "amountType": "Credit", "description": null, "movement": "No Movement", "entity": "Total Geography", "dataSource": "Journal Input"}]</pre>

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body.

```
{
  "scenario": "Actual",
  "createdOn": "2018-07-26 10:21:35.634",
  "modifiedBy": "epm_default_cloud_admin",
  "journalType": "Regular",
  "createdBy": "epm_default_cloud_admin",
  "balanceType": "Balanced",
  "postedBy": null,
  "journalLineItems",
  {
    "account": "Balance",
    "amount": "200",
    "amountType": "Credit",
    "description": null,
    "movement": "No Movement",
    "entity": "Total Geography",
    "dataSource": "Journal Input",
  }
  ],,
  "journalUrl": null,
  "year": "FY17",
  "description": "Journal Desc",
  "group": "grp1",
  "status": "Working",
  "label": "J2",
  "period": "Jan"
}
```

Copy Data

This REST API is used to execute a Copy Data job using the profile name. Before executing this job, you should create a Copy Data profile in Financial Consolidation and Close.

For details on this task, see [Using Copy Data Profiles](#).

This REST API returns the job id after starting the Copy Data job.

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/jobs

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Table 14-17 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with: v3	Path	Yes	None
application	The name of the application Get the application name by using the Get Applications API, for example, FCCS or TRCS. See Get Applications .	Path	Yes	None
jobName	Name of the job should be: EXECUTE PROFILE	String	Yes	None
jobType	Type of Job. Supported value: COPY_DATA	String	Yes	None
ProfileName	Name of the profile to use to copy data	String	Yes	None

Example of Request Body

```
{
  "jobType": "Copy_Data",
  "jobName": "Execute Profile",
  "parameters": {
    "ProfileName": "<ProfileName>",
  }
}
```

Response Body

Supported Media Types: application/json

Table 14-18 Parameters

Name	Description
type	Financial Consolidation and Close Application type, for example, FCCS
status	Status of the job: -1 =In progress; 0 = Success; 1 = Fail
details	In case of errors, details are published with the error string.
descriptiveStatus	The status of the job, such as Completed or Error
items	Collection of Notification categories
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type. Possible values: self

Example of Response Body:

The following shows an example of the response body in JSON format.

```
{
  "jobId": 8,
  "descriptiveStatus": "Processing",
}
```

```

"details": null,
"jobName": "Copy Data",
"status": -1,
"links": [
  {
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/<applicationName>/jobs/<JobId>", "rel": "self", "action": "GET"},
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/
    "rel": "job-details", "action": "GET"},
  }
]
}

```

Clear Data

This REST API is used to execute a Clear Data job using the profile name. Before executing this job, you should create a Clear Data profile in Financial Consolidation and Close.

For details on this task, see [Using Copy Data Profiles](#).

This REST API returns the job id after starting the job.

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/jobs

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 14-19 Clear Data

Name	Description	Type	Required	Default
api_version	Version of the API you are working with: v3	Path	Yes	None

Table 14-19 (Cont.) Clear Data

Name	Description	Type	Required	Default
application	The name of the application Get the application name by using the Get Applications API, for example, FCCS or TRCS. See Get Applications .	Path	Yes	None
jobType	Type of Job. Supported value: Clear_Data	String	Yes	None
ProfileName	The name of the profile to use to clear data	String	Yes	None

Example of request body

Example:

```
{
  "jobType": "Clear_Data",
  "jobName": "Execute Profile",
  "parameters": {
    "ProfileName": "<ClearData_01>",
  }
}
```

Response**Table 14-20 Parameters**

Name	Description
type	Financial Consolidation and Close Application type, for example, FCCS
status	Status of the job: -1 =In progress; 0 = Success; 1 = Fail
details	In case of errors, details are published with the error string.
descriptiveStatus	The status of the job, such as Completed or Error
items	Collection of Notification categories
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type. Possible values: self

Supported Media Types: application/json

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "jobId": 8,
  "descriptiveStatus": "Processing",
  "details": null,
  "jobName": "Clear Data",
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/<applicationName>/jobs/<JobId>", "rel": "self", "action": "GET"},
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/
      "rel": "job-details", "action": "GET"},
    ]
  }
}
```

Sample

```
{
  "jobId": 8,
  "descriptiveStatus": "Processing",
  "details": null,
  "jobName": "Clear Data",
  "status": -1,
  "links": [
    {
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/ Testapp /jobs/8",
      "rel": "self", "action": "GET"},
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/Testapp/jobs/8/details",
      "rel": "job-details", "action": "GET"}
    ]
  }
}
```

Deploy Form Templates

Enables you to deploy form templates that have been created in Financial Consolidation and Close.

For details on [Using Deploying a Form Template to a Data Collection Period](#).

This REST API returns the job id after starting the job.

REST Resource

POST /HyperionPlanning/rest/{api_version}/applications/{application}/fcmjobs

Required Roles

Service Administrator, Power User

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 14-21 Deploy Form Template

Name	Description	Type	Required	Default
api_version	Version of the API you are working with: v3	Path	Yes	None
application	The name of the application Get the application name by using the Get Applications API, for example, FCCS. See Get Applications .	Path	Yes	None
jobName	Name of the job that has been defined in Financial Consolidation and Close or no value for the job	String	No	None
jobType	Type of Job. Supported value: DEPLOY_FORM_TEMPLATES	String	Yes	None
CollectionIntervalName	The name of the collection interval to which the template should be deployed	String	Yes	None

Table 14-21 (Cont.) Deploy Form Template

Name	Description	Type	Required	Default
Parameter for frequency dimension 1	Dimension and member name of interval frequency dimension, to which the template should be deployed. This optional key value parameter depends on the number of frequency dimensions in the interval. The key should be dimension name and the value should be member name.	String	No	None
Parameter for frequency dimension 2	Dimension and member name of interval frequency dimension, to which the template should be deployed. This optional key value parameter depends on the number of frequency dimensions in the interval. The key should be dimension name and the value should be member name.	String	No	None

Table 14-21 (Cont.) Deploy Form Template

Name	Description	Type	Required	Default
Parameter for frequency dimension 3	Dimension and member name of interval frequency dimension, to which the template should be deployed. This optional key value parameter depends on the number of frequency dimensions in the interval. The key should be dimension name and the value should be member name.	String	No	None
Parameter for frequency dimension 4	Dimension and member name of interval frequency dimension, to which the template should be deployed. This optional key value parameter depends on the number of frequency dimensions in the interval. The key should be dimension name and the value should be member name.	String	No	None

Table 14-21 (Cont.) Deploy Form Template

Name	Description	Type	Required	Default
Template	<p>Names of the form templates to deploy. You can provide multiple parameters with the same key as "TEMPLATE" and this will be converted to a JSON array and passed as a single parameter in the JSON string.</p> <p>This is an optional parameter. However, if nothing is specified for the TEMPLATE key, this parameter is still present as an empty JSON Array in the JSON string, and all templates for the given interval will be deployed.</p> <p>Even if you give only one template name, this should still be passed as a JSON array.</p>	JSON Array	No	None
ResetWorkflows	<p>Either true or false. Optional parameter indicating whether all forms should be reset back to the first stage after redeploying.</p> <p>This parameter is also derived by the system based on changes to the template and collection, so the system-derived value can override the user specified value.</p>	Boolean	No	false

Examples of request body

Example 1:

```
{
  "jobType" : "DEPLOY_FORM_TEMPLATES",
  "parameters":
    {
      "CollectionIntervalName" : "Journal Collection
Interval",
      "Year" : "2020",
      "Period" : "July",
      "Product" : "Oracle EPM",
      "Consolidation" : "Entity Input",
      "Template" : [ "Template,1","Template 2" ],
      "ResetWorkflows" : "true"
    }
}
```

Example 2:

```
{
  "jobType" : "DEPLOY_FORM_TEMPLATES",
  "parameters":
    {
      "CollectionIntervalName" : "Loan Collection
Interval",
      "Year" : "2020",
      "Period" : "July",
      "Category" : "Oracle EPM",
      "Movement" : "Actual",
      "Template" : ["Template 3"]
    }
}
```

Example 3:

```
{
  "jobType" : "DEPLOY_FORM_TEMPLATES",
  "parameters":
    {
      "CollectionIntervalName" : "Default",
      "Year" : "2020",
      "Period" : "July",
      "Scenario" : "Actual",
      "Template" : ["Template 5","Template 6"],
      "ResetWorkflows" : "false",
    }
}
```

Example 4:

```
{
  "jobType" : "DEPLOY_FORM_TEMPLATES",
  "parameters":
    {
      "CollectionIntervalName" : "Custom Interval",
      "Year" : "2020",
      "Period" : "July",
      "Template" : ["Template 5","Template 6","Template
7","Template 8"]
    }
}
```

Response**Table 14-22 Parameters**

Name	Description
jobId	Financial Consolidation and Close job ID
descriptiveStatus	
details	Any additional details
Status	-1 = In progress; 0 = Success; 1 = Fail
items	Not applicable for this job type
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Possible values: self

Supported Media Types: application/json

JSON Output

The following is an example of the response body in JSON format.

```
{
  "jobId":100000000114040,
  "descriptiveStatus":",
  "detail":"In Progress",
  "status":-1,
  "items":null,
  "links":[
    {
      "rel":"self",
      "href":"https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3/
applications/FCCS/fcmjobs/100000000114040",
      "action":"GET"
    }
  ]
}
```

Deploy Task Manager Templates

Deploys a Task Manager template to provided year and period to create a new schedule.

This API executes the job based on the job type (TM_DEPLOY_TEMPLATE) provided as a JSON parameter. This is an asynchronous API and responds with a callback API to get the job status.

Parameters

The following table summarizes the client request:

Table 14-23 TM_DEPLOY_TEMPLATES Parameters

Name	Description	Type	Required	Default
jobType	Type of the Job, value for this Job is TM_DEPLOY_TEMPLATE	String	Yes	None
templateName	The name of the task manager template to be deployed	String	Yes	None
scheduleName	The name of the new schedule that will be created from the template	String	Yes	None
year	The member of the Year dimension where the template will be deployed	String	Yes	None
period	The member of the Period dimension where the template will be deployed	String	Yes	None
dayZeroDate	The Day Zero date used in creating the Schedule in a valid format	String	Yes	None
dateFormat	The date format for the Day Zero Date	String	No	YYY-MM-DD
orgUnit	The Organization Unit name	String	No	None

REST Resource

POST/HyperionPlanning/rest/cmapi/{api_version}/jobs

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request parameters specific to this job.

Table 14-24 Parameters

Name	Description	Type	Required	Default
api_version	It is a path parameter. Version of the API you are developing with for example, v1	String	Yes	None

Examples of Request Body

Example for Job Type: TM_DEPLOY_TEMPLATE

Example 1:

```
{
  "jobType": "TM_DEPLOY_TEMPLATE",
  "parameters": {
    "templateName": "Template1",
    "scheduleName": "scheduleA",
    "year": "2021",
    "period": "Jan",
    "dayZeroDate": "2021-01-01"
  }
}
```

Example 2:

```
{
  "jobType": "TM_DEPLOY_TEMPLATE",
  "parameters": {
```

```

        "templateName": "Template1",
        "scheduleName": "scheduleA",
        "year": "2021",
        "period": "Jan",
        "dayZeroDate": "2021-01-01",
        "orgUnit": "JPAC"
    }
}

```

Response

Supported Media Types: application/json

Table 14-25 Parameters

Name	Description
jobId	Job ID
descriptiveStatus	Any additional details
details	Message to the end user. In case of errors, details are published.
status	-1 – In Progress; 0 – Success; 1 – Fail
items	Not applicable for this job type
links	Detailed information about the link
rel	Possible values: self
href	Links to API call
action	The HTTP call type

JSON Output

The following is an example of the response body in JSON format.

```

{
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/cmapi/v1/jobs",
      "action": "POST"
    }
  ]
}

```

```
    },  
    {  
        "rel": "Job Status",  
        "href": "https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/  
cmapi/v1/jobs/14008",  
        "action": "GET"  
    }  
],  
"details": "In Process",  
"status": -1,  
"type": "TM",  
"link": {},  
"error": null,  
"items": []  
}
```

Update Task Status for Event Monitoring

This API is used to update the task status to "completed" based on an event monitoring integration type. For information on setting up integration type for Event Monitoring, see [Creating Custom Event Monitoring Integrations](#) in *Administering Financial Consolidation and Close* .

REST Resource

POST /HyperionPlanning/rest/cmapi/{api_version}/updateTasksForEventMonitoring

Required Roles

Service Administrator

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 14-26 Parameters

Name	Description	Type	Required	Version
api_version	Version of the API you are developing with	Path	Yes	v1

Example URL

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/cmapi/v1/updateTasksForEventMonitoring
```

Example Payload

With the following payload, the tasks based on custom event monitoring integration (name=customPeriodClose, eventName=custom.CloseProcess.period.close, integrationConnectionName= customAppConn) will be updated which are waiting for the specified parameters periodName=FY18 and ledgerID=123.

```
{
  "eventName" : "custom.CloseProcess.period.close",
  "integrationName" : "customPeriodClose",
  "integrationConnectionName" : "customAppConn",
  "parameters": [{
    "name": "periodName",
    "value": "FY18"
  }, {
    "name": "ledgerID",
    "value": "123"
  }]
}
```

Response

Supported Media Types: application/json

Example of a Successful Response

Http status code: 200

Example of an Error Response

Http status: 500

Execute an Enterprise Journals Job

This API will execute the job based on the job type provided as a JSON parameter. It is an asynchronous API. This API will respond with a callback API to get the Job status.

Table 14-27 Enterprise Journals Supported Job Types

Job Type	Description
EJ_EXPORT_JOURNALS	This job exports the data of all "Ready to Post" journals. It creates one CSV file per journal based on the target definition and mappings defined in Enterprise Journals and copies it to an EPM Automate-designated directory. The resulting file follows the naming pattern: <Year>_<Period>_<Journal_ID>.csv. After creating the export file, it updates the journal Post status to "Post in Progress".
EJ_SET_JOURNAL_STATUS	This job updates the Post status of a journal after importing the journal to the ERP system. A CSV file containing the results of the journal posting is provided. The file contains these columns: Year, Period, Journal ID, Posting Status (Posted/Failed), Message, Error Code, Error Message. The Message, Error Code and Error Message columns are optional. Posting status of the journals is updated based on the status provided in the file. Only journals with a Posting status of "Post in Progress" will be updated.
EJ_DEPLOY_TEMPLATES	This job is used to deploy finalized journal templates to new journal periods to create journals for year-period combinations, ensuring a consistent and repeatable journal collection process.

REST Resource

POST /HyperionPlanning/rest/ej/{api_version}/jobs

Required Roles

Job Type	Role
EJ_EXPORT_JOURNALS	Service Administrator
EJ_SET_JOURNAL_STATUS	Service Administrator
EJ_DEPLOY_TEMPLATES	Service Administrator, Power User

Request

Supported Media Types: application/json

Parameters

This table summarizes the request parameters that are generic to all jobs. The following tables describe parameters specific to individual rules.

Table 14-28 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None

Table 14-29 EJ_EXPORT_JOURNALS Parameters

Name	Description	Type	Required	Default
jobType	Type of the Job. Value for this job is EJ_EXPORT_JOURNALS.	String	Yes	None
filename	The name of the zip file to which each individual journal file is added	String	Yes	None
year	The year for which the journal is used for data collection	String	No	None
period	The period for which the journal is used for data collection	String	No	None

Table 14-30 EJ_SET_JOURNAL_STATUS Parameters

Name	Description	Type	Required	Default
jobType	Type of the Job. Value for this job is EJ_SET_JOURNAL_STATUS.	String	Yes	None

Table 14-30 (Cont.) EJ_SET_JOURNAL_STATUS Parameters

Name	Description	Type	Required	Default
filename	<p>Filename is the file in CSV format that contains the following information regarding the posting process:</p> <ul style="list-style-type: none"> • Year: Year of the Journal posted • Period: Period of the Journal posted • JournalID: Journal Identifier of Journal posted • Posting Status: Valid values are Posted or Failed. Failed status has a message and attached error list returned from ERP • Message: Optional. Can be used for error messages from ERP in case there is a posting failure <p>Note: Multiple Journal Posting statuses may be included as records in the CSV file.</p>	String	Yes	None

Table 14-31 EJ_DEPLOY_TEMPLATES Parameters

Name	Description	Type	Required	Default
jobType	Type of the Job. Value for this job is EJ_DEPLOY_TEMPLATES.	String	Yes	None
year	The year used for the journal	String	Yes	None
period	The period for the journal	String	Yes	None
Template	Names of journal templates that need to be deployed. You can specify multiple parameters with the same key as "TEMPLATE" in the command, and it will be converted into JSON array and passed as a single parameter in JSON string. If you only specify one template name, it is still passed as JSON array. This parameter is optional. If no value is specified in the command for the TEMPLATE key, then this parameter would still be present as an empty JSON array in JSON string, and all templates for the specified year and period will be deployed.	JSON Array	No	None

Table 14-31 (Cont.) EJ_DEPLOY_TEMPLATES Parameters

Name	Description	Type	Required	Default
ResetJournals	Value values: True or False. Optional. Indicates whether or not all journals need to be reset to the first stage after re-deployment. This parameter is also derived by the system based on changes to the template. The system-derived value can override the user-specified value.	Boolean	No	False

Examples of Request Body**Example for Job Type: EJ_EXPORT_JOURNALS**

Example 1:

```
{
  "jobType" : "EJ_EXPORT_JOURNALS",
  "parameters": {
    "Filename": "export.zip",
    "Year": "2020"
    "Period": "Jan"
  }
}
```

Example 2:

```
{
  "jobType" : "EJ_EXPORT_JOURNALS",
  "parameters": {
    "Filename": "export.zip",
  }
}
```

Example for Job Type: EJ_SET_JOURNAL_STATUS

```
{
  "jobType" : "EJ_SET_JOURNAL_STATUS",
  "parameters": {
    "Filename": "JournalsStatus.csv",
  }
}
```

```
    }  
}
```

Examples for Job Type: EJ_DEPLOY_TEMPLATES

Example 1:

```
{  
  "jobType" : "EJ_DEPLOY_TEMPLATES",  
  "parameters": {  
    "Year": "2020"  
    "Period": "July"  
    "Template": ["Template 1", "Template 2"],  
    "ResetJournals": "true"  
  }  
}
```

Example 2:

```
{  
  "jobType" : "EJ_DEPLOY_TEMPLATES",  
  "parameters": {  
    "Year": "2020"  
    "Period": "July"  
    "Template": ["Template 3"],  
  }  
}
```

Example 3:

```
{  
  "jobType" : "EJ_DEPLOY_TEMPLATES",  
  "parameters": {  
    "Year": "2020"  
    "Period": "July"  
    "Template": ["Template 5", "Template 6",  
"Template 7", "Template 8"],  
  }  
}
```

Example 4:

```
{  
  "jobType" : "EJ_DEPLOY_TEMPLATES",  
  "parameters": {  
    "Year": "2020"  
    "Period": "July"  
    "Template": [ ]  
  }  
}
```

Response

Supported Media Types: application/json

Table 14-32 Parameters

Name	Description
jobId	Enterprise Journals job identifier
descriptiveStatus	Additional details about the status
details	Message to the user. In cases where there are errors, the error details are published.
status	<ul style="list-style-type: none">-1: In Progress0: Success1: Failed
items	Not applicable for this job type
links	Detailed information about the link
rel	Valid value: self
href	Links to API call
action	The HTTP call type

Example of Response Body

The following is an example of the response body in JSON format.

```
{
  "jobId":14013,
  "details": "In Process",
  "status": -1,
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.
oraclecloud.com/HyperionPlanning/rest/ej/v1/jobs/14013",
      "action": "GET"
    }
  ],
  "error": null,
  "items": [ ],
  "type": EPM
}
```

Retrieve Enterprise Journals for Financial Consolidation and Close

Returns the list of Enterprise Journals ready to post.

This API works only for Financial Consolidation and Close.

REST Resource

GET /HyperionPlanning/rest/ej/{api_version}/ejournals

Required Roles

Service Administrator

Request URL with Optional Parameters

GET /HyperionPlanning/rest/ej/{api_version}/ejournals?
Year=2018&Period=Jan&PostingStatus=ReadyToPost

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

By default, only ReadyToPost journals are listed, however you can provide any valid value for the PostingStatus parameter to get corresponding journals.

Table 14-33 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with: v1	Path	Yes	None
Year	Year for which to list journals, for example, Year=2018. If you do not specify a Year and Period, journals from all Years and Periods are listed.	Query	No	All Years
Period	Period for the journal, for example: Period=Jan. If you do not specify a Year and Period, journals from all Years and Periods are listed. If you specify only Year or only Period, journals from all Years and Periods are listed.	Query	No	All Periods
Postingstatus	Posting status for the journal, for example: PostingStatus=ReadyToPost Valid values are NotPosted, ReadyToPost, PostInProgress, Failed, Posted	Query	No	ReadyToPost

Response

Supported Media Types: application/json

Example of Response Body

The following shows an example of the response body.

```
{
  "items": [
    {
      "year": 2020
      "period": "Jan",
      "journalId": "100000001",
      "instanceId": "100000000008821",
      "link": {
        "rel": "content",
        "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejournals/100000000008821",
        "action": "GET"
      }
    }
  ],
  "year": 2020
  "period": "Jan",
  "journalId": "100000002",
  "instanceId": "100000000008822",
  "link": {
    "rel": "content",
    "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejournals/100000000008822",
    "action": "GET"
  }
},
"link": {
  "rel": "self",
  "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejournals",
  "action": "GET"
}
"error": null,{
"type": "EPM"
}
```

Retrieve Enterprise Journal Content for Financial Consolidation and Close

Returns journal content for the instance identifier provided as Path parameter. Each item in the items list represents a line item of the journal.

This API works only for Financial Consolidation and Close.

REST Resource

```
GET /HyperionPlanning/rest/ej/{api_version}/ejournals/
{instanceId}
```

Required Roles

Service Administrator

Example of Request URL

```
GET /HyperionPlanning/rest/ej/v1/ejournals/100000000008821
```

Supported Media Types: application/json

Table 14-34 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
instanceId	Identifier for the journal for which you want to retrieve journal content	Path	Yes	None

Example of Response Body

The following shows an example of the response body.

```
{
  "year": 2018
  "period": "Jan",
  "journalId": "100000001",
  "instanceId": "100000000008821",
  "items": [
    {
      "Status Code": "NEW",
      "Ledger ID": "LNR 12000",
      "Journal Source": "EPM_EJ",
      "Journal Category": "Adjustment",
      "Currency Code": "EUR",
      "Segment 1": "100091",
      "Entered Debit Amount": "19800.00",
      "Entered Credit Amount": "0.00"
    },
    {
      "Status Code": "NEW",
      "Ledger ID": "LNR 12000",
      "Journal Source": "EPM_EJ",
      "Journal Category": "Adjustment",
      "Currency Code": "EUR",
```

```

    "Segment 1": "100092",
    "Entered Debit Amount": "0.00",
    "Entered Credit Amount": "19800.00"
  },
  {
    "Status Code": "NEW",
    "Ledger ID": "LNR 12000",
    "Journal Source": "EPM_EJ",
    "Journal Category": "Adjustment",
    "Currency Code": "EUR",
    "Segment 1": "100093",
    "Entered Debit Amount": "34900.00",
    "Entered Credit Amount": "0.00"
  },
  {
    "Status Code": "NEW",
    "Ledger ID": "LNR 12000",
    "Journal Source": "EPM_EJ",
    "Journal Category": "Adjustment",
    "Currency Code": "EUR",
    "Segment 1": "1000943",
    "Entered Debit Amount": "0.00",
    "Entered Credit Amount": "34900.00"
  }
],
  "links": [
    {
      "rel": "self",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejjournals/100000000008821",
      "action": "GET"
    },
    {
      "rel": "update_posting_status",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejjournals/100000000008821
/poststatus",
      "action": "POST"
    }
  ],
  "error": null,
  "type": "EPM"
}

```

Retrieve Enterprise Journal Content by Year and Period for Financial Consolidation and Close

Returns Enterprise journal content for the provided Year, Period, and Journal ID. This API can be used if a user wants to query an Enterprise Journal without knowing the journal identifier.

This API works only for Financial Consolidation and Close.

REST Resource

```
GET /HyperionPlanning/rest/ej/{api_version}/ejjournalcontent?
Year={year}&Period={period}&JournalId={journalId}
```

Required Roles

Service Administrator

Example of Request URL

```
GET /HyperionPlanning/rest/ej/v1/ejournalcontent?
Year=2018&Period=Jan&JournalId=100000001
```

Supported Media Types: application/json

Table 14-35 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
Year	Year for which to list journal, for example, Year=2018	Query	Yes	None
Period	Period for the journal, for example: Period=Jan	Query	Yes	None
JournalId	Identifier for the journal for which you want to retrieve journal content	Query	No	If journal ID is not specified, it lists the content of all journals for the specified Year and Period.

Example of Response Body

The following shows an example of the response body.

```
{
  "year": 2018
  "period": "Jan",
  "journalId": "100000001",
  "instanceId": "100000000008821",
  "items": [
    {
      "Status Code": "NEW",
      "Ledger ID": "LNR 12000",
      "Journal Source": "EPM_EJ",
      "Journal Category": "Adjustment",
      "Currency Code": "EUR",
      "Segment 1": "100091",
      "Entered Debit Amount": "19800.00",
```

```

    "Entered Credit Amount": "0.00"
  },
  {
    "Status Code": "NEW",
    "Ledger ID": "LNR 12000",
    "Journal Source": "EPM_EJ",
    "Journal Category": "Adjustment",
    "Currency Code": "EUR",
    "Segment 1": "100092",
    "Entered Debit Amount": "0.00",
    "Entered Credit Amount": "19800.00"
  },
  {
    "Status Code": "NEW",
    "Ledger ID": "LNR 12000",
    "Journal Source": "EPM_EJ",
    "Journal Category": "Adjustment",
    "Currency Code": "EUR",
    "Segment 1": "100093",
    "Entered Debit Amount": "34900.00",
    "Entered Credit Amount": "0.00"
  },
  {
    "Status Code": "NEW",
    "Ledger ID": "LNR 12000",
    "Journal Source": "EPM_EJ",
    "Journal Category": "Adjustment",
    "Currency Code": "EUR",
    "Segment 1": "1000943",
    "Entered Debit Amount": "0.00",
    "Entered Credit Amount": "34900.00"
  }
],
"links": [
  {
    "rel": "self",
    "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejjournals/100000000008821",
    "action": "GET"
  },
  {
    "rel": "update_posting_status",
    "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/
rest/ej/v1/ejjournals/100000000008821
/poststatus",
    "action": "POST"
  }
],
"error": null,
"type": "EPM"
}

```

Update Enterprise Journal Posting Status for Financial Consolidation and Close

Updates the Enterprise Journal Posting status. After journal content has been read and the import to ERP begins, this API must be invoked to update the status to PostInProgress, and after completion of posting, the status should be updated with either Posted or Failed. Error items are read-only if the Posting status is Failed.

Note:

The maximum character limit of the following fields are:

- journalBatch – 255 characters
- errorMessage – 255 characters
- errorCode – 20 characters
- entryId – 500 characters

This API works only for Financial Consolidation and Close.

REST Resource

```
POST /HyperionPlanning/rest/ej/{api_version}/ejjournals/
{instanceId}/poststatus
```

Required Roles

Service Administrator

Sample Request

```
POST /HyperionPlanning/rest/ej/v1/ejjournals/100000000008821/
Posted
```

Supported Media Types: application/json

Table 14-36 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with: v1	Path	Yes	None
instanceId	The ID of the journal for which you want to update Posting status.	Path	Yes	None
poststatus	The journal status. Valid values are PostInProgress, Failed, and Posted.	Path	Yes	None

Example of Post In Progress:

```
{
  "year": "2019",
  "period": "Jan",
  "journalId": "10000000012",
  "status": "PostInProgress"
}
```

Example of an Error:

```
{
  "year": "2019",
  "period": "Jan",
  "journalId": "1000000001",
  "status": "Failed",
  "errorItems": [
    {
      "errorCode": "EF04",
      "errorMessage": "The account combination is invalid.",
      "entryId": "Journal Ent 1"
    },
    {
      "errorCode": "EU07",
      "errorMessage": "Accounting period is not open for the ledger.",
      "entryId": "Journal Ent 2"
    }
  ]
}
```

Example When Posted:

```
{
  "year": "2021",
  "period": "Jan",
  "journalId": "10000000003",
  "status": "Posted",
  "journalBatch": "111720201001-LNR11432"
}
```

Response

```
{
  "detail": "Journal 2021 Jan 10000000001 is not in 'Post In Progress' or 'Ready to Post' status.",
  "status": 400,
  "message": "oracle.apps.epm.sdm.model.common.SDMMModelException: Journal 2021 Jan 10000000001 is not in 'Post In Progress' or 'Ready to Post' status.",
  "localizedMessage": "oracle.apps.epm.sdm.model.common.SDMMModelException: Journal 2021 Jan 10000000001 is not in 'Post In Progress' or 'Ready to Post' status.",
  "suppressed": []
}
```

 **Note:**

The maximum character limit of the following fields are:

- journalBatch – 255 characters
- errorMessage – 255 characters
- errorCode – 20 characters
- entryId – 500 characters

Enterprise Profitability and Cost Management REST APIs

Related Topics

- [URL Structure for Enterprise Profitability and Cost Management](#)
- [Getting API Versions for Enterprise Profitability and Cost Management](#)
You can get information on REST API versions using REST resources.
- [Getting Information About a Specific REST API Version for Enterprise Profitability and Cost Management](#)
Returns information about a specific REST API version for Enterprise Profitability and Cost Management.
- [Calculate Model](#)
Runs the calculation on a given point of view in a selected cube. This API supports batch calculation with multiple POVs.
- [Clear Data By Point Of View](#)
Clears the data for a given point of view in a selected cube.
- [Copy Data by Point of View](#)
Copies data from a source to a destination point of view in a selected cube.
- [Delete Point of View](#)
Deletes the data associated with a point of view from the calculation cube.
- [Validate Model](#)
Automates the calculation process for validating a model.

URL Structure for Enterprise Profitability and Cost Management

Use the following URL structure to access the Enterprise Profitability and Cost Management REST resources:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/  
HyperionPlanning/rest/{api_version}/{path}
```

Where:

api_version —API version you are developing with. The current REST API version for Enterprise Profitability and Cost Management is v3.

path —Identifies the resource

Getting API Versions for Enterprise Profitability and Cost Management

You can get information on REST API versions using REST resources.

Important: The version number is case-sensitive. For example, if the version number is listed as v3 with a lowercase v, you cannot enter the version number with a capital V as in this incorrect example, V3, which would result in an error. Instead, you must enter the version number with a lowercase v as in this correct example: v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using REST APIs requires prerequisites. See [Prerequisites](#).

There are two sets of REST APIs relevant for Enterprise Profitability and Cost Management.

- Planning REST APIs (See [Getting API Versions for Planning](#)).
- Enterprise Profitability and Cost Management-specific REST APIs. (See [Getting Information About a Specific REST API Version for Enterprise Profitability and Cost Management](#)).

Getting Information About a Specific REST API Version for Enterprise Profitability and Cost Management

Returns information about a specific REST API version for Enterprise Profitability and Cost Management.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /HyperionPlanning/rest/{api_version}

Request

Supported Media Types: application/json

Parameters

The following table summarizes the client request.

Table 15-1 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are working with, such as V3	Path	Yes	None

Response Body

Supported Media Types: application/json

The following table summarizes the response parameters.

Table 15-2 Parameters

Attribute	Description
version	v3
lifecycle	Lifecycle of the resource, active or deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "version": "v3",
  "lifecycle": "active",
  "isLatest": true,
  "links": [{
    "rel": "canonical",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v3"
  }, {
    "rel": "predecessor-version",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/HyperionPlanning/rest/v2"
  }]
}
```

Calculate Model

Runs the calculation on a given point of view in a selected cube. This API supports batch calculation with multiple POVs.

This is an asynchronous call, so use the job status URI to determine whether the operation is complete.

This API is version v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrators

REST Resource

POST /HyperionPlanning/rest/v3/applications/{AppName}/jobs/

Request

Supported Media Types: application/json

The following table summarizes the client request parameters specific to this job

Table 15-3 Parameters

Name	Description	Type	Required	Default
jobType	Calculation	Payload	Yes	None
jobName	Name of the job Example: "Calculation"	Payload	Yes	None
povDelimiter	Delimiter used in POV values Example: "povDelimiter": ":"	Payload	Yes	:: (Double Colon)
povName	Name of the POV to calculate. You can pass one or more POVs separated by a comma (,). Example: "sourcePOVName": "FY16:May:Actual:Working"	Payload	Yes	None
modelName	Name of the model to calculate Example: "modelName": "10 Actuals Allocation Process"	Payload	Yes	None
executionType	(ALL_RULES RULESET_SUBSET SINGLE_RULE RUN_FROM_RULE STOP_AFTER_RULE) Identifies the rule execution type <ul style="list-style-type: none"> If executionType=ALL_RULES, rule related parameters are not required. If executionType=SINGLE_RULE RUN_FROM_RULE STOP_AFTER_RULE, then provide ruleName only. If executionType=RULESET_SUBSET, then provide values for rulesetSeqNumStart and rulesetSeqNumEnd. Example: "executionType": "ALL_RULES"	Payload	Yes	None
ruleName	Name of the single rule to run Example: "ruleName": "Utilities Expense Adjustment"	Payload	No	None
rulesetSeqNumStart	Sequence number of the first rule in the rule set to run Example: "rulesetSeqNumStart:1"	Payload	No	None
rulesetSeqNumEnd	Sequence number of the last rule in the rule set to run Example: "rulesetSeqNumStart:10"	Payload	No	None

Table 15-3 (Cont.) Parameters

Name	Description	Type	Required	Default
clearCalculatedData	(True False) Specifies whether to clear existing calculations Example: "clearCalculatedData": "true",	Payload	No	False
executeCalculations	(True False) Specifies whether to run calculations Example: "executeCalculations": "true"	Payload	No	False
optimizeForReporting	(True False) Specifies whether to optimize the calculation process for reporting When running multiple concurrent calculation jobs, set optimizeReporting=true for all jobs. Only the last job to complete will perform the aggregation, which avoids redundant processing and prevents running jobs from slowing down. Set optimizeReporting=false only when necessary to save processing time; for example, when running a single rule or a sequential series of several POVs. Example: "optimizeForReporting": "false",	Payload	No	False
captureDebugScripts	(True False) Specifies whether to generate debug scripts Example: "captureDebugScripts": "false"	Payload	No	False
comment	Comment to describe the job	Payload	No	None

Sample Payload:

```
{
  "jobType": "Calculation",
  "jobName": "Calculation",
  "parameters": {
    "povDelimiter": ":",
    "povName": "FY21:Jan:Actual:Working",
    "modelName": "10 Actuals Allocation Process",
    "executionType": "ALL_RULES",
    "rulesetSeqNumStart": "1",
    "rulesetSeqNumEnd": "1",
    "clearCalculatedData": "true",
    "executeCalculations": "true",
    "optimizeForReporting": "true",
    "captureDebugScripts": "false"
  }
}
```

Response Body

Supported Media Types: application/json

Table 15-4 Parameters

Name	Description
jobId	ID of the job that is created
jobName	Name of the job
details	In case of errors, details are published with the error string
status	See Migration Status Codes .
links	Detailed information about the link
href	Links to the API call
action	HTTP call type
rel	Can be self and/or Job-details. If set to Job Status, you can use the href to get the status of the job.
data	Parameters as key value pairs passed in the request

Examples of Response Body:

The examples show the response body in JSON format.

Example 1: Job in progress

```
{
  "jobId": 26,
  "jobName": "Calculation",
  "status": -1,
  "descriptiveStatus": "Processing",
  "details": null,
  "links": [
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26",
      "action": "GET",
      "rel": "self",
    },
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26/details",
      "action": "GET",
      "rel": "job-details "
    }
  ]
}
```

Example 2: Job status with no errors

```
{
  "jobId": 26,
  "jobName": "Calculation",
  "status": 0,
  "descriptiveStatus": "Success",
  "details": null
  "links": [
    {
      "rel": "self"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26",
      "action": "GET",
    },
    {
      "rel": "job-details"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26/details",
      "action": "GET",
    }
  ]
}
```

Clear Data By Point Of View

Clears the data for a given point of view in a selected cube.

This is an asynchronous call, so use the job status URI to determine whether the operation is complete.

This API is version v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrators

REST Resource

POST /HyperionPlanning/rest/v3/applications/{AppName}/jobs/

Request

Supported Media Types: application/json

The following table summarizes the client request parameters specific to this job.

Table 15-5 Parameters

Name	Description	Type	Required	Default
jobType	Clear POV	Form	Yes	None
jobName	Name of the job Example: "Clear POV"	Form	Yes	None
povDelimiter	Delimiter used in POV values Example: "povDelimiter":":":	Form	Yes	:: (Double Colon)
povName	Name of the POV to clear Example: "povName":":to FY21:Jan:Actual:Working"	Form	Yes	None
cubeName	Name of the cube on which clear operation is to be executed Example: "cubeName":":PCM_CLC"	Form	Yes	PCM_CLC
clearInput	(True False) Specifies whether to clear input data Example: "clearInput":":true"	Form	Yes	None
clearAllocatedValues	(True False) Specifies whether to clear allocated data Example: "clearAllocatedValues":":true"	Form	Yes	None
clearAdjustmentValues	(True False) Specifies whether to clear adjustment data Example: "clearAdjustmentValues":":true"	Form	Yes	None

Sample Payload:

```
{
  "jobType": "Clear POV",
  "jobName": "Clear POV",
  "parameters": {
    "povDelimiter": ":",
    "povName": "FY21:Jan:Actual:Working",
    "cubeName": "PCM_CLC",
    "clearInput": "true",
    "clearAllocatedValues": "true",
    "clearAdjustmentValues": "true"
  }
}
```

Response Body

Supported Media Types: application/json

Table 15-6 Parameters

Name	Description
jobId	ID of the job that is created
jobName	Name of the job
details	In case of errors, details are published with the error string
status	See Migration Status Codes .
links	Detailed information about the link
href	Links to the API call
action	HTTP call type
rel	Can be self and/or Job-details. If set to Job Status, you can use the href to get the status of the job.
data	Parameters as key value pairs passed in the request

Examples of Response Body:

The examples show the response body in JSON format.

Example 1: Job in progress

```
{
  "jobId": 26,
  "jobName": "Clear POV",
  "status": -1,
  "descriptiveStatus": "Processing",
  "details": null,
  "links": [
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
BksML40/jobs/26",
      "action": "GET",
      "rel": "self",
    },
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
BksML40/jobs/26/details",
      "action": "GET",
      "rel": "job-details "
    }
  ]
}
```

Example 2: Job status with no errors

```
{
  "jobId": 26,
  "jobName": " Copy POV",
```

```

    "status": 0,
    "descriptiveStatus": "Success",
    "details": null
    "links": [
      {
        "rel": "self"
        "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/jobs/26",
        "action": "GET",
      },
      {
        "rel": "job-details"
        "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/jobs/26/details",
        "action": "GET",
      }
    ]
  }
}

```

Copy Data by Point of View

Copies data from a source to a destination point of view in a selected cube.

This is an asynchronous call, so use the job status URI to determine whether the operation is complete.

This API is version v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrators

REST Resource

POST /HyperionPlanning/rest/v3/applications/{AppName}/jobs/

Request

Supported Media Types: application/json

The following table summarizes the client request parameters specific to this job.

Table 15-7 Parameters

Name	Description	Type	Required	Default
jobType	Copy POV	Form	Yes	None
jobName	Name of the job Example: "Copy POV"	Form	Yes	None
povDelimiter	Delimiter used in POV values Example: "povDelimiter":":":	Form	Yes	:: (Double Colon)

Table 15-7 (Cont.) Parameters

Name	Description	Type	Required	Default
sourcePOVName	Name of the source POV Example: "sourcePOVName": "FY21:Jan:Actual:Working "	Form	Yes	None
destinationPOVName	Name of the destination POV Example: "destinationPOVName": "FY21:Feb:Actual:Working"	Form	Yes	None
copyType	(ALL_DATA INPUT) specifies the data to copy from the source Example: "copyType": "ALL_DATA"	Form	Yes	None
sourceCubeName	Name of the source Essbase cube Example: "sourceCubeName": "PCM_CLC"	Form	Yes	None
destCubeName	Name of the destination Essbase cube Example: "destCubeName": "PCM_CLC"	Form	Yes	None

Sample Payload:

```
{
  "jobType": "Copy POV",
  "jobName": "Copy POV",
  "parameters": {
    "povDelimiter": ":",
    "sourcePOVName": "FY21:Jan:Actual:Working",
    "destinationPOVName": "FY21:Feb:Actual:Working",
    "sourceCubeName": "PCM_CLC",
    "destCubeName": "PCM_CLC",
    "createDestPOV": "true",
    "copyType": "ALL_DATA"
  }
}
```

Response Body

Supported Media Types: application/json

Table 15-8 Parameters

Name	Description
jobId	ID of the job that is created
jobName	Name of the job
details	In case of errors, details are published with the error string
status	See Migration Status Codes .
links	Detailed information about the link

Table 15-8 (Cont.) Parameters

Name	Description
href	Links to the API call
action	HTTP call type
rel	Can be self and/or Job-details. If set to Job Status, you can use the href to get the status of the job.
data	Parameters as key value pairs passed in the request

Examples of Response Body:

The examples show the response body in JSON format.

Example 1: Job in progress

```
{
  "jobId": 26,
  "jobName": "Copy POV",
  "status": -1,
  "descriptiveStatus": "Processing",
  "details": null,
  "links": [
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26",
      "action": "GET",
      "rel": "self",
    },
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26/details",
      "action": "GET",
      "rel": "job-details "
    }
  ]
}
```

Example 2: Job status with no errors

```
{
  "jobId": 26,
  "jobName": " Copy POV",
  "status": 0,
  "descriptiveStatus": "Success",
  "details": null
  "links": [
    {
      "rel": "self"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
```

```

applications/jobs/26",
      "action": "GET",
    },
  {
    "rel": "job-details"
    "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26/details",
    "action": "GET",
  }
}

```

Delete Point of View

Deletes the data associated with a point of view from the calculation cube.

This is an asynchronous call, so use the job status URI to determine whether the operation is complete.

This API is version v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrators

REST Resource

POST /HyperionPlanning/rest/v3/applications/{AppName}/jobs/

Request

Supported Media Types: application/json

The following table summarizes the client request parameters specific to this job.

Table 15-9 Parameters

Name	Description	Type	Required	Default
jobType	Delete POV	Form	Yes	None
jobName	Name of the job Example: "Delete POV"	Form	Yes	None
povDelimiter	Delimiter used in POV values Example: "povDelimiter": ":"	Form	Yes	:: (Double Colon)
povName	Name of the POV to delete Example: "povName": "FY21:Jan:Actual:Working"	Form	Yes	None

Sample Payload:

```

{
  "jobType": "Delete POV",

```

```

    "jobName": "Delete POV",
    "parameters": {
      "povDelimiter": ":",
      "povName": "FY21:Jan:Actual:Working"
    }
  }
}

```

Response Body

Supported Media Types: application/json

Table 15-10 Parameters

Name	Description
jobId	ID of the job that is created
jobName	Name of the job
details	In case of errors, details are published with the error string
status	See Migration Status Codes .
links	Detailed information about the link
href	Links to the API call
action	HTTP call type
rel	Can be self and/or Job-details. If set to Job Status, you can use the href to get the status of the job.
data	Parameters as key value pairs passed in the request

Examples of Response Body:

The examples show the response body in JSON format.

Example 1: Job in progress

```

{
  "jobId": 26,
  "jobName": "Delete POV",
  "status": -1,
  "descriptiveStatus": "Processing",
  "details": null,
  "links": [
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26",
      "action": "GET",
      "rel": "self",
    },
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/
applications/BksML40/jobs/26/details",
      "action": "GET",
    }
  ]
}

```

```

        "rel": "job-details "
      }
    ]
  }

```

Example 2: Job status with no errors

```

{
  "jobId": 26,
  "jobName": " Delete POV",
  "status": 0,
  "descriptiveStatus": "Success",
  "details": null
  "links": [
    {
      "rel": "self"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26",
      "action": "GET",
    },
    {
      "rel": "job-details"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26/details",
      "action": "GET",
    }
  ]
}

```

Validate Model

Automates the calculation process for validating a model.

This is an asynchronous call, so use the job status URI to determine whether the operation is complete.

Any validation failures are written to file with file name provided in the parameters, and can be accessed from Inbox/Outbox Explorer.

This API is version v3.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Required Roles

Service Administrators

REST Resource

POST /HyperionPlanning/rest/v3/applications/{AppName}/jobs/

Request

Supported Media Types: application/json

The following table summarizes the client request parameters specific to this job.

Table 15-11 Parameters

Name	Description	Type	Required	Default
jobType	Validate Model	Form	Yes	None
jobName	Name of the job Example: "Validate Model"	Form	Yes	None
modelName	Name of the model to validate Example: "modelName":"10 Actuals Allocation Process"	Form	Yes	None
fileName	Name of the output file to which all the validations (if any) will be written Example: "fileName": "results.txt"	Form	Yes	None

Sample Payload:

```
{
  "jobType":"Validate Model",
  "jobName":"Validate Model",
  "parameters":{
    "modelName":"10 Actuals Allocation Process",
    "fileName":"results.txt"
  }
}
```

Response Body

Supported Media Types: application/json

Table 15-12 Parameters

Name	Description
jobId	ID of the job that is created
jobName	Name of the job
details	In case of errors, details are published with the error string
status	See Migration Status Codes .
links	Detailed information about the link
href	Links to the API call
action	HTTP call type
rel	Can be self and/or Job-details. If set to Job Status, you can use the href to get the status of the job.

Table 15-12 (Cont.) Parameters

Name	Description
data	Parameters as key value pairs passed in the request

Examples of Response Body:

The examples show the response body in JSON format.

Example 1: Job in progress

```
{
  "jobId": 26,
  "jobName": "Validate Model",
  "status": -1,
  "descriptiveStatus": "Processing",
  "details": null,
  "links": [
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
BksML40/jobs/26",
      "action": "GET",
      "rel": "self",
    },
    {
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
BksML40/jobs/26/details",
      "action": "GET",
      "rel": "job-details "
    }
  ]
}
```

Example 2: Job status with no errors

```
{
  "jobId": 26,
  "jobName": "validate Model",
  "status": 0,
  "descriptiveStatus": "Success",
  "details": null
  "links": [
    {
      "rel": "self"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
jobs/26",
      "action": "GET",
    },
    {
      "rel": "job-details"
      "href": "http://<epm_host>/HyperionPlanning/rest/v3/applications/
```

```
jobs/26/details",  
    "action": "GET",  
  }  
]  
}
```

16

Profitability and Cost Management REST APIs

The following topics provide information on the Profitability and Cost Management REST APIs.

URL Structure for Profitability and Cost Management

Use the following URL structure to access the Profitability and Cost Management REST resources:

```
https://<SERVICE_NAME>-  
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/{api_version}/  
{path}
```

Where:

api_version —API version you are developing with. The current REST API version for is V1.

path —Identifies the resource

Getting API Versions for Profitability and Cost Management REST APIs

You can manage versions using the set of REST resources summarized in the following table.

Before using the REST resources, you must understand how to access the REST resources and other important concepts. See [Implementation Best Practices for EPM Cloud REST APIs](#). Using this REST API requires prerequisites. See [Prerequisites](#).

Table 16-1 Manage Versions of Profitability and Cost Management APIs

Task	Request	REST Resource
Get API Versions for Profitability and Cost Management REST APIs	GET	/epm/rest/
Get Information about a Specific API Version for Profitability and Cost Management	GET	/epm/rest/{apiVersion}

Get API Versions for Profitability and Cost Management REST APIs

Returns information about which versions are available and supported. Multiple versions might be supported simultaneously.

**Note:**

An API version is always supported even when deprecated.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /epm/rest/

Request

Supported Media Types: application/json

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 16-2 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
items	Version of the API you are developing with
version	The version, such as v1
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "isLatest": false,
    "lifecycle": "deprecated",
    "version": "11.1.2.4.000",
    "links": [{
```

```

        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/11.1.2.4.000",
        "rel": "canonical"
    }, {
        "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1",
        "rel": "successor-version"
    }
  ]
}, {
  "isLatest": true,
  "lifecycle": "active",
  "version": "v1",
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1",
    "rel": "canonical"
  }, {
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/11.1.2.4.000",
    "rel": "predecessor-version"
  }
  ],
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/11.1.2.4.000",
    "rel": "canonical"
  }, {
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1",
    "rel": "current"
  }
  ]
}
}

```

Java Sample – GetRestAPIVersionsInfo.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void getRestAPIVersionsInfo() throws Exception {
    String urlString = String.format("%s/epm/rest", serverUrl);
    String response = executeRequest(urlString, "GET", null,
"application/json");
    System.out.println("Response String : " + response);
    JSONObject jsonObj = new JSONObject(response);
    JSONArray itemsArray = jsonObj.getJSONArray("items");
    System.out.println("Details : " + itemsArray.toString());
}

```

cURL Sample – GetRestAPIVersionInfo.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#)

```
funcGetRestAPIVersionInfo()
{
    url=$SERVER_URL/epm/rest/$API_VERSION
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"
    status=$?
    echo "status code :$status"
    output='cat response.txt'
    if [ $status == 200 ]; then
        echo "Version $API_VERSION details :"
        count='echo $output | jq '.links | length''
        i=0
        while [ $i -lt $count ]; do
            echo "Service : " 'echo $output | jq '.links['$i'].rel''
            echo "URL : " 'echo $output | jq '.links['$i'].href''
            echo "Action : " 'echo $output | jq '.links['$i'].action''
            echo ""
            i='expr $i + 1'
        done
    else
        error='echo $output'
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – GetRestAPIVersionsInfo.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def getRestAPIVersionsInfo() {
    def url;
    def response;

    try {
        url = new URL(serverUrl + "/epm/rest");
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }

    response = executeRequest(url, "GET", null, "application/json");
    def object = new JsonSlurper().parseText(response)

    if(object != null) {
        def items = object.items
        println "Rest API Versions Info : " + items
    } else {
        println "Error occurred while fetching rest api
versions details"
```

```
}
}
```

Get Information about a Specific API Version for Profitability and Cost Management

Returns details for a specific REST API version for Profitability and Cost Management.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

GET /epm/rest/

Request

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 16-3 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as V1	Path	Yes	None

Response

Supported Media Types: application/json

Parameters

The following table summarizes the parameters.

Table 16-4 Parameters

Name	Description
version	The version, such as v1
lifecycle	Possible values: active, deprecated
isLatest	Whether this resource is the latest, true or false

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "items": [{
    "version": "v1",
```

```

    "lifecycle": "active",
    "isLatest": true,
    "links": [{
      "rel": "canonical",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1",
    }, {
      "rel": "predecessor-version",
      "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1",
    }
  ]
}, {
  "links": [{
    "rel": "current",
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1"
  }
]
}

```

Apply Data Grants

Applies data grants for a given Profitability and Cost Management application.

This API submits a job to remove all existing data grants in Essbase and recreate them with the latest information from the application. It can also be used to repair data grants if there are any issues.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/jobs/
applyDataGrants

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-5 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as v1	Path	Yes	None
application	Name of the application to create	Path	Yes	None

Example URL

```

https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/BksML30/jobs/applyDataGrants

```


Response Body

Supported Media Types: application/json

Table 16-6 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_LoadData_D20160118T051020_ba8_1
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_ApplyDataGrants_D20220511T114653_b85",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_ApplyDataGrants_D20220511T114653_b85",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – applyDataGrants.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void applyDataGrants() throws Exception {
```

```
        String urlString = "%s/epm/rest/%s/applications/%s/jobs/  
applyDataGrants";  
        executeJob(urlString, "POST", null);  
    }  
}
```

cURL Sample – ApplyDataGrants.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```
funcApplyDataGrants() {  
  
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/  
applyDataGrants  
    funcExecuteRequest "POST" $url "application/json"  
  
    output=`cat response.txt`  
    status=`echo $output | jq '.status'`  
    if [ $status == -1 ]; then  
        echo "Started Data Grants successfully"  
        funcGetStatus "GET"  
    else  
        error=`echo $output | jq '.details'`  
        echo "Error occurred. " $error  
    fi  
    funcRemoveTempFiles "respHeader.txt" "response.txt"  
  
}
```

Groovy Sample – ApplyDataGrants.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def applyDataGrants() {  
    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/  
applications/" + appName + "/jobs/applyDataGrants";  
    def url;  
  
    try {  
        url = new URL(urlString)  
    } catch (MalformedURLException e) {  
        println "Malformed URL. Please pass valid URL"  
        System.exit(0);  
    }  
}
```

```
executeJob(url, "POST", null);
}
```

Create File-Based Application

Creates an application using a flat file using a REST API.

Required Roles

Service Administrator

REST Resource

POST /epm/rest/{api_version}/fileApplications/{application}

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-7 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application to create	Path	Yes	None
description	User comment for this application	Payload	Yes	None
ruleDimensionName	Rule dimension name	Payload	Yes	None
balanceDimensionName	Balance dimension name	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/
rest/v1/fileApplications/BksML12
```

```
{"description":
"description", "ruleDimensionName": "Rule", "balanceDimensionName": "Balance"}
```

Response Body

Supported Media Types: application/json

Table 16-8 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_LoadData_D20160118T051020_ba8_1
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs

Table 16-8 (Cont.) Parameters

Name	Description
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_UpdateDimensions_D20220513T062046_c61",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_UpdateDimensions_D20220513T062046_c61",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – CreateFlatFileApplication.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void createFlatFileApplication() throws Exception {

    JSONObject json = new JSONObject();
    json.put("description", "Flat file based application");
    json.put("ruleDimensionName", "Rule");
    json.put("balanceDimensionName", "Balance");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
fileApplications/" + "BksML13";
    String response = executeRequest(urlString, "POST",
json.toString(), "application/json");

    JSONObject jsonObj = new JSONObject(response);
```

```

int resStatus = jsonObj.getInt("status");

if(resStatus == 0) {
    System.out.println("Application created successfully");
} else {
    System.out.println("Application creation failed");
}
}

```

cURL Sample – CreateFlatFileApplication.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

```

funcCreateFlatFileApplication() {
    description="Flat file based application";
    ruleDimensionName="Rule"
    balanceDimensionName="Balance"

    param="{\"description\": \"${description}\", \"ruleDimensionName\": \"${ruleDimensionName}\", \"balanceDimensionName\": \"${balanceDimensionName}\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/fileApplications/BksML13
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Application created successfully"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – CreateFlatFileApplication.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def createFlatFileApplication() {

    JSONObject json = new JSONObject();
    json.put("description", "Flat file based application");
    json.put("ruleDimensionName", "Rule");
    json.put("balanceDimensionName", "Balance");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
fileApplications/" + "BksML13";

```

```

def url;

try {
    url = new URL(urlString)
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}

String response = executeRequest(url, "POST", json.toString(),
"application/json")

JSONObject jsonObj = new JSONObject(response);
int resStatus = jsonObj.getInt("status");

if(resStatus == 0) {
    println "Application created successfully"
} else {
    println "Application creation failed"
}
}

```

Deploy ML Cube

Deploys or redeploys the calculation cube for a selected Profitability and Cost Management application.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/jobs/ledgerDeployCubeJob

Request


Supported Media Types: application/json

The following table summarizes the client request.

Table 16-9 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to deploy the cube	Path	Yes	None
isKeepData	Specify whether to preserve existing data	Payload	Yes	None

Table 16-9 (Cont.) Parameters

Name	Description	Type	Required	Default
isReplaceCube	Specify whether to replace existing cube	Payload	Yes	None
<div style="border: 1px solid #0070c0; padding: 10px; width: fit-content; margin: 10px auto;">  Note: Both <i>isKeepData</i> and <i>isReplaceCube</i> cannot be true at the same time. </div>				
isRunNow	Run now or schedule for later. (Schedule for later is not currently supported.)	Payload	Yes	None
comment	Any user comments	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcx>.oraclecloud.com/epm/
rest/v1/applications/{applicationName}/jobs/ledgerDeployCubeJob
```

```
{"isKeepData":"true","isRunNow":"true","comment":"Test M1 Deploy"}
```

Response Body

Supported Media Types: application/json

Table 16-10 Parameters

Name	Description
details	In case of errors, details are published with the error string
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type":"Profitability",
  "status":-1,
}
```

```

    "statusMessage": "In Progress",
    "details": "BksML30_DeployCube_D20220511T114550_da1",
    "links": [
      {
        "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_DeployCube_D20220511T114550_da1",
        "action": "GET",
        "rel": "Job Status"
      }
    ]
  }
}

```

Java Sample – DeployCube.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void deployCube() throws Exception {
    JSONObject json = new JSONObject();
    json.put("isKeepData", true);
    json.put("isReplaceCube", false);
    json.put("isRunNow", true);
    json.put("comment", "Cube deployment");

    String urlString = "%s/epm/rest/%s/applications/%s/jobs/
ledgerDeployCubeJob";
    executeJob(urlString, "POST", json.toString());
}

```

cURL Sample – DeployCube.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcDeployCube() {
    comment="Cube deployment Curl"

    param="{\"isKeepData\": \"false\", \"isReplaceCube\": \"true\", \"isRunNow\": \"true\", \"comment\": \"${comment}\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/ledgerDeployCubeJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then

```



```
        echo "Started Deploying Cube successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – DeployCube.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def deployCube() {

    JSONObject json = new JSONObject();
    json.put("isKeepData", true);
    json.put("isReplaceCube", false);
    json.put("isRunNow", true);
    json.put("comment", "Cube deployment");

    def url;
    def response;

    try {
        url = new URL(serverUrl + "/epm/rest/" + apiVersion + "/"
applications/" + appName + "/jobs/ledgerDeployCubeJob")
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }

    executeJob(url, "POST", json.toString());
}
```

Copy ML POV Data

Copies model artifacts and data from a Source POV combination to a Destination POV combination for any application. Use with Management Ledger applications.

Required Roles

Service Administrator, Power User

REST Resource

```
POST /epm/rest/{api_version}/applications/{application}/povs/
{srcPOVMemberGroup}/jobs/copyPOVJob/{destPOVMemberGroup}
```

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-11 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to deploy the cube	Path	Yes	None
srcPOVMemberGroup	Source POV member group, such as 2014_January_Actual	Path	Yes	None
destPOVMemberGroup	Destination POV member group, such as 2014_March_Actual	Path	Yes	None
isManageRule	Whether to copy the program rule details	Payload	Yes	None
isInputData	Whether to copy input data Note: Do not set the value of this parameter to true if the value of isAllData or isAllInputData is set to true.	Payload	Yes	None
modelViewName	Whether to copy a slice of data from source POV to destination POV	Payload	No	Nothing copied
isAllInputData	Whether to copy all input data at the NoRule member, including AdjustmentIn/Out	Form	No	False
isAllData	Whether to copy all POV data	Form	No	False
createDestPOV	Whether to create the destination POV if it does not already exist	Payload	Yes	None
nonEmptyTupleEnabled	Specifies whether to enable the Non Empty Tuple (NET). Valid values are <i>true</i> and <i>false</i> . In some cases, the default <code>nonEmptyTupleEnabled=true</code> does not perform well when copying the Essbase data. In those cases, use <code>nonEmptyTupleEnabled=false</code> to improve performance.	Payload	No	True
stringDelimiter	String delimiter for POV group members	Payload	Yes	None

Example URL and Payload

```

https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/LM1T2/povs/2014_January_Actual/jobs/copyPOVJob/
2014_March_Actual

{"isManageRule":"true","isInputData":"true","modelViewName":"Operating
Expenses","createDestPOV":"true","nonEmptyTupleEnabled":"true","stringDelim
iter":"_"}

```

Response Body

Supported Media Types: application/json

Table 16-12 Parameters

Name	Description
details	Task ID, such as LM1T2_LM1T2_CopyMLPOV_D20160113T065943_75b_1
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_CopyMLPOV_D20220511T114800_7ad",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_CopyMLPOV_D20220511T114800_7ad",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – CopyPOV.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void copyPOVData() throws Exception {

    JSONObject json = new JSONObject();
    String
    modelAndViewName = "Operating Expenses";
    json.put("isManageRule", true);
    json.put("isInputData", true);
    json.put("modelAndViewName", modelAndViewName);
    json.put("createDestPOV", true);
    json.put("stringDelimiter", "_");
}
```

```

String sourcePovGroupMember = "2014_January_Actual";
String destPovGroupMember = "2014_December_Actual";

String urlString = "%s/epm/rest/%s/applications/%s/povs/" +
sourcePovGroupMember.trim().replaceAll(" ", "%20")
                    + "/jobs/copyPOVJob/" +
destPovGroupMember.trim().replaceAll(" ", "%20");
executeJob(urlString, "POST", json.toString());
}

```

cURL Sample – CopyPOV.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcCopyPOVData() {
    stringDelimiter="_";    modelViewName="Operating Expenses";
    destPovGroupMember="2014_December_Actual";

    param="{\"isManageRule\": \"true\", \"isInputData\": \"true\", \"modelViewName\": \"$modelViewName\", \"createDestPOV\": \"true\", \"stringDelimiter\": \"$stringDelimiter\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/
    povs/$POV_GROUP_MEMBER/jobs/copyPOVJob/$destPovGroupMember
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Copying POV successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Java Sample – CopyPOV.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void copyPOVData() throws Exception {

    JSONObject json = new JSONObject();                String
    modelViewName = "Operating Expenses";
    json.put("isManageRule", true);
}

```

```

        json.put("isInputData", true);
        json.put("modelName", modelName);
        json.put("createDestPOV", true);
        json.put("stringDelimiter", "_");

        String sourcePovGroupMember = "2014_January_Actual";
        String destPovGroupMember = "2014_December_Actual";

        String urlString = "%s/epm/rest/%s/applications/%s/povs/" +
sourcePovGroupMember.trim().replaceAll(" ", "%20")
                                + "/jobs/copyPOVJob/" +
destPovGroupMember.trim().replaceAll(" ", "%20");
        executeJob(urlString, "POST", json.toString());
    }

```

Enable File-Based Application

Enables an application using a flat file.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/fileApplications/{application}/enableApplication

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-13 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application to enable	Path	Yes	None

Example URL

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/fileApplications/BksML12/enableApplication`

Response Body

Supported Media Types: application/json

Table 16-14 Parameters

Name	Description
details	Task ID, such as BksM112_BksM112_EnableApplication_D2 0160113T075011_53c_1
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_EnableApplication_D20220511T114947_65c",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_EnableApplication_D20220511T114947_65c",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – EnableApplication.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void enableApplication() throws Exception {
    String urlString = "%s/epm/rest/%s/fileApplications/%s" + "/"
enableApplication";
```

```
        executeJob(urlString, "POST", null);
    }
}
```

cURL Sample – EnableApplication.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```
funcEnableApplication() {
    url=$SERVER_URL/epm/rest/$API_VERSION/fileApplications/$APP_NAME/
enableApplication
    funcExecuteRequest "POST" $url "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Enabling Application successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – EnableApplication.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def enableApplication() {

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
fileApplications/" + appName + "/enableApplication";
    def url;

    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    executeJob(url, "POST", null);
}
```

Essbase Data Load for Profitability and Cost Management

Loads input data to an Essbase application.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/jobs/essbaseDataLoadJob

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-15 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to load the data	Path	Yes	None
clearAllDataFlag	Whether to clear existing data (true) or not (false)	Payload	Yes	None
dataLoadValue	Possible values are ADD_EXISTING_VALUES or OVERWRITE_EXISTING_VALUES	Payload	Yes	None
dataFileName	Name of the data file already present in the inbox folder	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/BksML12/jobs/essbaseDataLoadJob

{"clearAllDataFlag":"true","dataLoadValue":"OVERWRITE_EXISTING_VALUES","dataFileName":"input.txt"}
```

Response Body

Supported Media Types: application/json

Table 16-16 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_LoadData_D20160118T051020_ba8_1
status	See Migration Status Codes
statusMessage	Message about the status, such as Success

Table 16-16 (Cont.) Parameters

Name	Description
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_LoadData_D20220511T114654_8bf",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_LoadData_D20220511T114654_8bf",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – EssbaseDataLoad.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void loadData() throws Exception {

    JSONObject json = new JSONObject();
    json.put("clearAllDataFlag", false);
    json.put("dataLoadValue", "ADD_EXISTING_VALUES");
    json.put("dataFileName", "BksML12C.txt");

    String urlString = "%s/epm/rest/%s/applications/%s/jobs/
essbaseDataLoadJob";
    executeJob(urlString, "POST", json.toString());

}
```

cURL Sample – EssbaseDataLoad.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```
funcLoadData() {
    dataLoadValue="ADD_EXISTING_VALUES"
    dataFileName="BksML12C.txt"

    param="{\"clearAllDataFlag\": \"false\", \"dataLoadValue\": \"$dataLoadValue\", \"dataFileName\": \"$dataFileName\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/essbaseDataLoadJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Loading Data successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – EssbaseDataLoad.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def loadData() {

    JSONObject json = new JSONObject();
    json.put("clearAllDataFlag", false);
    json.put("dataLoadValue", "ADD_EXISTING_VALUES");
    json.put("dataFileName", "BksML12C.txt");

    def url;
    def response;

    try {
        url = new URL(serverUrl + "/epm/rest/" + apiVersion +
"/applications/" + appName + "/jobs/essbaseDataLoadJob")
    } catch (MalformedURLException e) {
```

```

        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    executeJob(url, "POST", json.toString());
}

```

Export Template for Profitability and Cost Management

Exports Profitability and Cost Management applications as a template into the Outbox.

Required Roles

Service Administrator, Power User

REST Resource

POST/epm/rest/{api_version}/applications/{application}/jobs/templateExportJob?fileName={fileName}

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-17 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application	Path	Yes	None
fileName	Name of the template zip file to be exported to the outbox folder	Query	Yes	None

Note:

If the file name is the same as an existing file name, this will override content in existing file.

Example URL and Payload

https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/BksML30/jobs/templateExportJob{"fileName":"testFile"}

Response Body

Supported Media Types: application/json

Table 16-18 Parameters

Name	Description
details	Task ID, such as BksML30_ExportTemplate_D20180201T210316_a80
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_ExportTemplate_D20220511T114738_16e",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_ExportTemplate_D20220511T114738_16e",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – ExportTemplate.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void exportTemplate() throws Exception {
    String fileName = applicationName + "_Template_Export_File";
```

```

        JSONObject json = new JSONObject();
        json.put("fileName", fileName);

        String urlString = "%s/epm/rest/%s/applications/%s/jobs/
templateExportJob";
        executeJob(urlString, "POST", json.toString());
    }

```

cURL Sample – ExportTemplate.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcExportTemplate() {
    fileName=$APP_NAME+"_Template_Export_File"
    param="{\"fileName\": \"${fileName}\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/
templateExportJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Exporting successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – ExportTemplate.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def exportTemplate() {
    String fileName = appName + "_Template_Export_File";

    JSONObject json = new JSONObject();
    json.put("fileName", fileName);

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
applications/" + appName + "/jobs/templateExportJob";

    def url;

```

```

try {
    url = new URL(urlString)
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}
executeJob(url, "POST", json.toString());
}

```

Export Query Results

Exports query results for a given query or exports all Essbase data into a file in the Outbox.

When exporting all Essbase data, there is an option for writing the output in columnar format, and columnar-formatted data can be filtered by level-0 dimension members. This API triggers a job that can be monitored in the Job Library.

Required Roles

Service Administrator, Power User, User, Viewer

REST Resource

POST /epm/rest/{api_version}/applications/{application}/jobs/exportQueryResultsJob

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-19 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as v1	Path	Yes	None
application	Name of the application	Path	Yes	None
exportOnlyLevel0Flg	Whether to export only Level0 data; values are true or false	Payload	No	false
fileName	Name of the query output file to be exported into the Outbox folder	Payload	Yes	None
fileOutputOptions	File output options. Available options are: <ul style="list-style-type: none"> ZIP_ONLY ZIP_AND_TEXT TEXT_ONLY 	Payload	No	ZIP_ONLY

Table 16-19 (Cont.) Parameters

Name	Description	Type	Required	Default
queryName	Query name from the Profitability and Cost Management application When queryName has a value, results for the given query are exported; exportOnlyLevel0Flg is considered if it is included. When queryName is blank or not included, data for the entire application is exported. In this case, exportOnlyLevel0Flg is ignored.	Payload	No	None
roundingPrecision	The rounding precision (decimal places) for exported data. (Note: Applies only if queryName is also used.)	Payload	No	2
dataFormat	Select the output format as native Essbase format, or as columnar format. Values are NATIVE or COLUMNAR. With the COLUMNAR option, all Essbase data is exported, so the queryName parameter is ignored. Data can be filtered using the memberFilters parameter.	Payload	No	NATIVE
The following parameters are only considered when dataFormat=COLUMNAR.				
memberFilters	Accepts a JSON formatted string for dimension and respective level-0 member format. For example: <pre>{\"Dim1\": [\"Mem1\"], \"Dim2\": [\"Mem21\", \"Mem22\"]}</pre>	Payload	No	None
includeHeader	Adds dimension names as column headers. Values are true or false.	Payload	No	true
delimiter	Character used to separate dimension members in the results file; must be enclosed in double quotes.	Payload	No	Space
keepDuplicateMemberFormat	When this parameter is set to true, prints member names in Essbase duplicate member format, such as [Account]@[Accoun1]. If set to false, only the member name is printed.	Payload	No	true

Example URL and Payload

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/Ex3F3/jobs/exportQueryResultsJob
```

```
{\"queryName\": \"Profitability - Product\", \"fileName\": \"ProfitabilityProduct_03232016.txt\", \"exportOnlyLevel0Flg\": \"true\", \"roundingPrecision\": \"3\"}
```

Response Body

Supported Media Types: application/json

Table 16-20 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_ExportQueryResults_D20160323T024820_f73_1
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_ExportQueryResults_D20220511T114843_935",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_ExportQueryResults_D20220511T114843_935",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – ExportQueryResult.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void exportTemplate() throws Exception {
    String fileName = applicationName + "_Template_Export_File";

    JSONObject json = new JSONObject();
    json.put("fileName", fileName);
}
```



```

        String urlString = "%s/epm/rest/%s/applications/%s/jobs/
templateExportJob";
        executeJob(urlString, "POST", json.toString());
    }

```

cURL Sample – ExportQueryResult.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcExportQueryResult() {
    queryName="Profitability - Product";
    fileName=$APP_NAME+"_"+queryName+"_Query_Result"

    param="{\"queryName\": \"${queryName}\", \"fileName\": \"${fileName}\", \"exportOnlyLevel0Flg\": \"false\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/
exportQueryResultsJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Exporting successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – ExportQueryResult.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def exportQueryResult() {
    String queryName = "Profitability - Product";
    String fileName = appName + "_" + queryName + "_Query_Result";

    JSONObject json = new JSONObject();
    json.put("queryName", queryName);
    json.put("fileName", fileName);
    json.put("exportOnlyLevel0Flg", false);
}

```

```

        String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
        applications/" + appName + "/jobs/exportQueryResultsJob";

        def url;

        try {
            url = new URL(urlString)
        } catch (MalformedURLException e) {
            println "Malformed URL. Please pass valid URL"
            System.exit(0);
        }
        executeJob(url, "POST", json.toString());
    }

```

Generate Program Documentation Report

Generates a Program Documentation report for a given Profitability and Cost Management point of view.

The report is generated in the profitoutbox folder with the name HPCMMLProgramDocumentationReport_{AppName}_{POV}.pdf. The file can be downloaded using File Explorer.

Required Roles

Service Administrator, Power User, User, or Viewer

REST Resource

GET epm/rest/{api_version}/applications/{application}/povs/{POV}/programDocumentationReport?
queryParameter={"fileType":"PDF","useAlias":"true"}

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-21 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as v1	Path	Yes	None
application	Name of the application for which to create the report	Path	Yes	None
pov	The POV for which to create the report, for example, FY17_JUN_Actual_Working	Path	Yes	None
fileType	The file format to use for the report, PDF, XML, WORD, EXCEL, or HTML	Query	No	PDF
useAlias	Boolean value to specify whether to use aliases in the report, true or false	Query	No	false

Example URL with fileType set to PDF and useAlias set to true:

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/
rest/v1/applications/BksML30/povs/2016_January_Actual/
programDocumentationReport?queryParameter={"fileType":"PDF","useAlias":"true"}
```

Response Body

Supported Media Types: application/json

Table 16-22 Parameters

Name	Description
details	Program Documentation report name, such as HPCMMLProgramDocumentationReport_BksML30_2016_January_Actual.pdf, and report status
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type":"Profitability",
  "status":0,
  "statusMessage":"Success",
  "details":"Program Documentation report
HPCMMLProgramDocumentationReport_BksML30_2016_January_Actual.pdf generated
successfully in the Outbox folder."
}
```

Java Sample – GeneratePrgrmDocReport.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void generatePrgrmDocReport() throws Exception {

    JSONObject json = new JSONObject();
    json.put("fileType", "PDF");
    json.put("useAlias", false);
    json.put("stringDelimiter", "_");
```

```

String povGroupMember = "2016_January_Actual";

String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
applications/" + applicationName + "/povs/" +
povGroupMember.trim().replaceAll(" ", "%20") + "/"
programDocumentationReport";
    urlString = urlString + "?" + "queryParameter=" +
json.toString();

String response = executeRequest(urlString, "GET", null,
"application/json");
JSONObject jsonObj = new JSONObject(response);
int resStatus = jsonObj.getInt("status");

if(resStatus == 0) {
    System.out.println("Program Documentation Report Generated
Successfully");
}
String details = jsonObj.getString("details");
System.out.println(details);
}

```

cURL Sample – GeneratePrgDocReport.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcGeneratePrgDocReport() {
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/
povs/$POV_GROUP_MEMBER1/programDocumentationReport
    echo $url
    curl -G "$url" --data-urlencode
'queryParameter={"fileType":"PDF","stringDelimiter":"_","useAlias":"fals
e"}' -u "$USERNAME:$PASSWORD" -o "response.txt" -D "respHeader.txt"
    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    echo $status
    if [ $status == 0 ]; then
        echo "Program Documentation Report generated successfully"
        message=`echo $output | jq '.details'`
        echo $message
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – GeneratePrgrmDocReport.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def generateProgramDocReportJob() {
    JSONObject json = new JSONObject();
    json.put("fileName", "2016JanActual.pdf");
    json.put("fileType", "PDF");
    json.put("useAlias", false);
    json.put("stringDelimiter", "_");

    String povGroupMember = "2016_January_Actual";
    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/applications/"
    + appName + "/povs/" + povGroupMember.trim().replaceAll(" ", "%20") + "/
    jobs/programDocReportJob";

    def url;
    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    executeJob(url, "POST", json.toString());
}
```

Generate Program Documentation Report - Run as a Job

Submits a job to generate a Program Documentation report for a given Profitability and Cost Management point of view.

The report is generated in the profitoutbox folder with the name as fileName parameter value or HPCMMLProgramDocumentationReport_{AppName}_{POV}.pdf as default. The file can be downloaded using File Explorer or by using the EPM Automate downloadfile command.

Required Roles

Service Administrator, Power User, User, or Viewer

REST Resource

POST

```
/epm/rest/{api_version}/applications/<applicationName>/povs/<povName>/jobs/
programDocReportJob
```

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-23 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with, such as v1	Path	Yes	None
applicationName	Name of the application for which to create the report	Path	Yes	None
povName	The POV for which to create the report, for example, FY17_JUN_Actual_Working	Path	Yes	None
fileType	The file format to use for the report, PDF, XML, WORD, EXCEL, or HTML	Request Payload	No	PDF
fileName	Name of the output file	Request Payload	No	HPCMMLProgramDocumentationReport_<AppName>_POV.pdf
subsetStart	Rule set starting sequence number to specify a range of rule sets to include in the report	Request Payload	No	None
subsetEnd	Rule set ending sequence number to specify a range of rule sets to include in the report	Request Payload	No	None
useAlias	Boolean value to specify whether to use aliases in the report, true or false	Request Payload	No	False
stringDelimiter	POV Dimension members separator	Request Payload	No	"_"

Example URL

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/<applicationName>/povs/<povName>/jobs/programDocReportJob
```

Request Payload

```
{
  "fileName": "FY12ActualReport.pdf",
  "fileType": "PDF",
  "subsetStart": "1",
  "subsetEnd": "6",
  "useAlias": false,
  "stringDelimiter": "_"
}
```

Response Body

Supported Media Types: application/json

Table 16-24 Parameters

Name	Description
details	Program Documentation report name, such as HPCMMLProgramDocumentationReport_BksML30_2016_January_Actual.pdf, and report status

Table 16-24 (Cont.) Parameters

Name	Description
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "Program Documentation report 2016JanActual1.pdf generated
successfully in the Outbox folder.",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_ProgramDocumentation_D20220511T115113_52a",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – GeneratePrgrmDocReport.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void generateProgramDocReportJob() throws Exception {
    JSONObject json = new JSONObject();
    json.put("fileName", "2016JanActual1.pdf");
    json.put("fileType", "PDF");
    json.put("subsetStart", "1");
    json.put("subsetEnd", "6");
    json.put("useAlias", false);
    json.put("stringDelimiter", "_");
    String povGroupMember = "2016_January_Actual";
```

```

String urlString = serverUrl + "/epm/rest/" + apiVersion +
"/applications/" + applicationName + "/povs/" +
povGroupMember.trim().replaceAll(" ", "%20") +
"/jobs/programDocReportJob";
executeJob(urlString, "POST", json.toString());
}

```

cURL Sample – GeneratePrgDocReport.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcProgramDocReportJob() {
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/
povs/$POV_GROUP_MEMBER1/jobs/programDocReportJob
    stringDelimiter="_";

param="{\"fileName\": \"2016JanActual.pdf\", \"fileType\": \"PDF\", \"subsetStart\": \"1\", \"subsetEnd\": \"6\", \"useAlias\": \"false\", \"stringDelimiter\": \"$stringDelimiter\"}"

    echo $param
funcExecuteRequest "POST" $url $param "application/json"

output=`cat response.txt`
status=`echo $output | jq '.status'`
if [ $status == -1 ]; then
    echo "Started program doc report generation"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – GeneratePrgrmDocReport.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def generateProgramDocReportJob() {
    JSONObject json = new JSONObject();
    json.put("fileName", "2016JanActual");
    json.put("fileType", "PDF");
    json.put("subsetStart", "1");
    json.put("subsetEnd", "6");
}

```



```

json.put("useAlias", false);
json.put("stringDelimiter", "_");
String urlString = serverUrl + "/epm/rest/" + apiVersion +
    "/applications/" + appName + "/povs/" +
    povGroupMember.trim().replaceAll(" ", "%20") +
    "/jobs/programDocReportJob";
def url;
try {
    url = new URL(urlString)
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}
executeJob(url, "POST", json.toString());
}

```

Import Template for Profitability and Cost Management

Imports a template zip file as an application from the inbox.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/jobs/templateImportJob

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-25 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application	Path	Yes	None
description	User comment for the application	Payload	Yes	None
fileName	Name of the template zip file to be imported from the inbox folder	Payload	Yes	None
isApplicationOverwrite	Whether to override an application if one already exists with same name. Values are <i>true</i> or <i>false</i> .	Payload	Yes	None

Example URL and Payload

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/Ex3F3/jobs/templateImportJob`

```

{"description":"description","fileName":"
testFile12345.zip","isApplicationOverwrite":"true"}

```

Response Body

Supported Media Types: application/json

Table 16-26 Parameters

Name	Description
details	Task ID, such as TD_ae61e427d9ab4d6f99e3b87378fa1c94
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_ImportTemplate_D20220511T114059_d3b",
  "links": [
    {
      "href": "http:// :// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_ImportTemplate_D20220511T114059_d3b",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – ImportTemplate.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void importTemplate() throws Exception {

    JSONObject json = new JSONObject();
    json.put("description", "Import Template");
    json.put("instanceName", "PROFITABILITY_WEB_APP");
    json.put("essApplicationServer", "EssbaseCluster-1");
    json.put("sharedServicesProject", "EssbaseCluster-1");
    json.put("applicationType", "Management Ledger");
    json.put("fileName", "HPCM_BksML12_20160128_200053.zip");
    json.put("isApplicationOverwrite", true);

    String urlString = "%s/epm/rest/%s/applications/%s/jobs/
templateImportJob";
    executeJob(urlString, "POST", json.toString());

}
```

cURL Sample – ImportTemplate.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```
funcImportTemplate() {
    description="Import Template through Curl Sample"
    instance="PROFITABILITY_WEB_APP"
    essAppServer="EssbaseCluster-1"
    sharedServicesProject="EssbaseCluster-1"
    applicationType="Management Ledger"
    fileName="PCM_BksML12_20160413_042937.zip"
    isApplicationOverwrite="true"

    param="{\"description\": \"${description}\", \"instanceName\": \"${instance}\", \"ess
ApplicationServer\": \"${essAppServer}\", \"sharedServicesProject\": \"${sharedServ
icesProject}\", \"applicationType\": \"${applicationType}\", \"fileName\": \"${fileNa
me}\", \"isApplicationOverwrite\": \"${isApplicationOverwrite}\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/
templateImportJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started importing successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – ImportTemplate.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def importTemplate() {  
  
    JSONObject json = new JSONObject();  
    json.put("description", "Import Template");  
    json.put("instanceName", "PROFITABILITY_WEB_APP");  
    json.put("essApplicationServer", "EssbaseCluster-1");  
    json.put("sharedServicesProject", "EssbaseCluster-1");  
    json.put("applicationType", "Management Ledger");  
    json.put("fileName", "BksML12_Template.zip");  
    json.put("isApplicationOverwrite", true);  
  
    def url;  
    def response;  
  
    try {  
        url = new URL(serverUrl + "/epm/rest/" + apiVersion +  
"/applications/" + appName + "/jobs/templateImportJob")  
    } catch (MalformedURLException e) {  
        println "Malformed URL. Please pass valid URL"  
        System.exit(0);  
    }  
    println "URL : " + url  
    println "Payload : " + json.toString()  
    executeJob(url, "POST", json.toString());  
}
```

Merge Slices for Profitability and Cost Management

Merges all incremental data slices into the main database slices.

Optionally, removes the Essbase cells with zero values to make the cube compact.

Required Roles

Service Administrator, Power User

REST Resource

POST/epm/rest/{api_version}/applications/{application}/jobs/mergeSlices

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-27 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the Profitability and Cost Management application	Path	Yes	None
removeZeroCells	If "true", removes cells with zero values	Path	No	"false"

Request URI Example

`https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/BksML30/jobs/mergeSlices`

Request Payload:

```
{
  "removeZeroCells": "true"
}
```

Response Body

Supported Media Types: application/json

Table 16-28 Parameters

Name	Description
details	In case of errors, details are published with the error string.
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be self and/or Job Status. If set to Job Status, you can use the href to get the status of the import operation.
data	Parameters as key value pairs passed in the request

Example of Response Body

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_MERGE_CUBE_D20220511T115052_771",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/"
```

```

jobs/ChecktaskStatusJob/BksML30_MERGE_CUBE_D20220511T115052_771",
  "action": "GET",
  "rel": "Job Status"
}
]
}

```

Optimize ASO Cube

Optimizes the performance of queries for data extraction by creating aggregate views in ASO cubes for Profitability and Cost Management applications.

This command allows you to perform query optimization operations on ASO cubes in cases where default aggregation is deemed insufficient to meet your data extraction or reporting needs because of large data size. The typical optimization process is as follows:

- Drop default and query-based aggregations.
- Start query tracking.
- Run sample queries from Profitability and Cost Management Query Manager, Oracle Smart View for Office (Windows), or Data Management, and any other MDX queries representative of the type of queries for which optimization is desired to train Essbase.
- Create aggregation based on optimized or default queries.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/v1/applications/{AppName}/jobs/optimizeASOCube

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-29 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
appName	Name of the application used to run Optimize ASO	Path	Yes	None

Table 16-29 (Cont.) Parameters

Name	Description	Type	Required	Default
type	Type of operation. Valid values are: <ul style="list-style-type: none"> • <code>clearAggregations</code> removes default and query-based views. • <code>createAggregations</code> creates default Essbase aggregate views. Use this option to perform default aggregation instead of query-based aggregation. • <code>startQueryTracking</code> starts query tracking. Use this option to allow Essbase to collect optimization information for creating query-based aggregations. • <code>stopQueryTracking</code> stops query tracking. Use this option to stop Essbase from collecting optimization information. Essbase continues to collect optimization information until you stop query tracking or stop Essbase.) • <code>createQBOAggregations</code> creates Essbase aggregate views based on the optimized queries that you run after enabling query tracking. 	Form	Yes	None

Response Body

Supported Media Types: `application/json`

Table 16-30 Parameters

Name	Description
details	In case of errors, details are published with the error string.
status	See Migration Status Codes
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Can be <code>self</code> and/or <code>Job Status</code> . If set to <code>Job Status</code> , you can use the <code>href</code> to get the status of the import operation.
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_OptimizeASOCube_D20220511T115135_55d",
```

```

    "links":[
      {
        "href":"http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_OptimizeASOCube_D20220511T115135_55d",
        "action":"GET",
        "rel":"Job Status"
      }
    ]
  }
}

```

Java Sample – OptimizeASOCube.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void optimizeASOCube() throws Exception {

    JSONObject json = new JSONObject();
    json.put("type", "createAggregations");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
    applications/" + applicationName+"/jobs/optimizeASOCube";
    executeJob(urlString, "POST", json.toString());

}

```

cURL Sample – OptimizeASOCube.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcOptimizeASOCube() {
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/jobs/
    optimizeASOCube
    param="{\"type\": \"createAggregations\"}"
    echo $param
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Optimize ASO Cube successfully"
        funcGetStatus "GET"
    else

```



```

        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – OptimizeASOCube.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def optimizeASOCube() {

    JSONObject json = new JSONObject();
    json.put("type", "createAggregations");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
    applications/" + appName + "/jobs/optimizeASOCube";

    def url;
    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }

    executeJob(url, "POST", json.toString());
}

```

Retrieve Task Status for Profitability and Cost Management

Displays the current status of the job process name.

Required Roles

Service Administrator, Power User

REST Resource

GET/epm/rest/{api_version}/applications/jobs/checkTaskStatusJob/{processName}

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-31 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
applications	Included in the path	Path	Yes	None
processName	The ID of the process or task flow for which to check the task status	Path	Yes	None

Response Body

Supported Media Types: application/json

The following table summarizes the response parameters.

Table 16-32 Parameters

Name	Description
processName	The Process Name or TaskflowId, such as RBkML1_ExportTemplate_D20160112T025419_836
task	Task name, such as ExportTemplate
status	Task status, such as Success

Example of Response Body

```
{
  "type": "Profitability",
  "links": [{
    "href": "https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
RBkML1_ExportTemplate_D20160112T025419_836",
    "action": "GET",
    "rel": "self"
  }],
  "status": 0,
  "details":
  "ExportTemplate=Success,RBkML1_ExportTemplate_D20160112T025419_836=Done",
  "statusMessage": "Success"
}
```

Run ML Calculations

Runs or clears calculations for a selected application. You can run calculations using rules in a model POV against data in a different data POV without copying rules.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/povs/{povGroupMember}/jobs/runLedgerCalculationJob

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-33 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to run calculations	Path	Yes	None
povGroupMember	The model POV group member from which the rules will be used for calculations, such as 2016_January_Actual If <i>dataPOVName</i> is not passed, <i>povGroupMember</i> is used as both model and data POV.	Path	Yes	None
dataPOVName	The data POV group member for which to run calculations, such as 2015_January_Actual <i>exeType=ALL_RULES</i> is the valid combination while using <i>dataPOVName</i> .	Payload	No	None
isClearCalculated	Whether to clear the calculation data, true or false	Payload	No	None
isRunNow	Whether to run now (true) or schedule for later (false); schedule for later is not currently supported	Payload	Yes	true
optimizeReporting	Whether to optimize for reporting (true) or not (false). When <i>optimizeReporting</i> is used, Profitability and Cost Management runs default aggregations on the Essbase cube when the calculation is complete. You can also run this setting by itself, which improves performance for queries and analytics. If you don't pass this parameter, its setting is assumed to be true ("About Optimizing for Reporting" in <i>Administering Profitability and Cost Management</i>).	Payload	No	true
subsetStart	Rule Set Starting Sequence Number	Payload	No	None
subsetEnd	Rule Set Ending Sequence Number	Payload	No	None
ruleName	Rule Name for a SINGLE_RULE option	Payload	No	None
ruleSetName	Rule Set Name for a SINGLE_RULE option	Payload	No	None

Table 16-33 (Cont.) Parameters

Name	Description	Type	Required	Default
exeType	The execution type specifies which rules to run; possible values are ALL_RULES, RULESET_SUBSET, SINGLE_RULE. Other parameters are required based on the exeType value: <ul style="list-style-type: none"> • exeType = ALL_RULES overrides all other options like subsetStart, subsetEnd, ruleSetName, and ruleName. • exeType = RULESET_SUBSET considers only subsetStart and subsetEnd. • exeType = SINGLE_RULE considers only ruleSetName and ruleName. 	Payload	Yes	None
comment	Use comment text, such as "This is run by user1"	Payload	No	None
stringDelimiter	String delimiter for POV group members, such as _	Payload	No	None

Example URL and Payload without Passing Data POV

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/{api_version}/
applications/{application}/povs/{povGroupMember}/jobs /
runLedgerCalculationJob

{"isClearCalculated":"true","isExecuteCalculations":"true","isRunNow":"true",
"optimizeReporting":"false","comment":"This is run by
user1","exeType":"ALL_RULES","stringDelimiter":"_"}
```

Example URL and Payload with Data POV Passed

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/{api_version}/
applications/{application}/povs/{povGroupMember}/jobs/
runLedgerCalculationJob

{"dataPOVName":"2015_January_Actual","isClearCalculated":"true","isExecute
Calculations":"true","isRunNow":"true","optimizeReporting":"false","comment":
"This is run by user1","exeType":"ALL_RULES","stringDelimiter":"_"}
```

Response Body

Supported Media Types: application/json

Table 16-34 Parameters

Name	Description
details	Task ID, such as BksML1_BksML1_RunCalcs_D20160113T070358_1da_1
status	See Migration Status Codes

Table 16-34 (Cont.) Parameters

Name	Description
statusMessage	Message about the status, such as In Progress
type	Profitability
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type": "Profitability",
  "status": -1,
  "statusMessage": "In Progress",
  "details": "BksML30_RunCalcs_D20220511T114716_a14",
  "links": [
    {
      "href": "http:// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_RunCalcs_D20220511T114716_a14",
      "action": "GET",
      "rel": "Job Status"
    }
  ]
}
```

Java Sample – RunCalculation.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void runCalculation() throws Exception {

    String subsetStart = null;
    String subsetEnd = null;
    String ruleName = null;
    String ruleSetName = null;

    JSONObject json = new JSONObject();
    json.put("isClearCalculated", true);
    json.put("isExecuteCalculations", true);
    json.put("isRunNow", true);
    json.put("comment", "Run Calculation");
}
```

```

        json.put("subsetStart", subsetStart);
        json.put("subsetEnd", subsetEnd);
        json.put("ruleName", ruleName);
        json.put("ruleSetName", ruleSetName);
        json.put("exeType", "ALL_RULES");
        json.put("stringDelimiter", "_");

        String povGroupMember = "2014_January_Actual";

        String urlString = "%s/epm/rest/%s/applications/%s/povs/" +
        povGroupMember.trim().replaceAll(" ", "%20")
                                + "/jobs/
runLedgerCalculationJob";
        executeJob(urlString, "POST", json.toString());
    }

```

cURL Sample – RunCalculation.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcRunCalculation() {
    subsetStart=""
    subsetEnd=""
    ruleName=""
    ruleSetName=""
    comment="Run Calculation Curl"
    exeType="ALL_RULES"
    stringDelimiter="_"

    param="{\"isClearCalculated\": \"true\", \"isExecuteCalculations\": \"true
\", \"isRunNow\": \"true\", \"comment\": \"${comment}\", \"subsetStart\": \"${su
bsetStart}\", \"subsetEnd\": \"${subsetEnd}\", \"ruleName\": \"${ruleName}\", \"r
uleSetName\": \"${ruleSetName}\", \"exeType\": \"${exeType}\", \"stringDelimite
r\": \"${stringDelimiter}\"}"
    url=${SERVER_URL}/epm/rest/${API_VERSION}/applications/${APP_NAME}/
povs/${POV_GROUP_MEMBER}/jobs/runLedgerCalculationJob
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Running Calc successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

```
}
```

Groovy Sample – RunCalculation.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def runCalculation() {

    String subsetStart = null;
    String subsetEnd = null;
    String ruleName = null;
    String ruleSetName = null;

    JSONObject json = new JSONObject();
    json.put("isClearCalculated", true);
    json.put("isExecuteCalculations", true);
    json.put("isRunNow", true);
    json.put("comment", "Run Calculation");
    json.put("subsetStart", subsetStart);
    json.put("subsetEnd", subsetEnd);
    json.put("ruleName", ruleName);
    json.put("ruleSetName", ruleSetName);
    json.put("exeType", "ALL_RULES");
    json.put("stringDelimiter", "_");

    String povGroupMember = "2014_January_Actual";
    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
applications/" + appName + "/povs/"
                                + povGroupMember.trim().replaceAll("
", "%20") + "/jobs/runLedgerCalculationJob";

    def url;

    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }
    executeJob(url, "POST", json.toString());
}
```

Run ML Clear POV

Clears model artifacts and data from a POV combination for any application.

Required Roles

Service Administrator, Power User

REST Resource

POST /epm/rest/{api_version}/applications/{application}/povs/
{povGroupMember}/jobs /clearPOVJob

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-35 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to run calculations	Path	Yes	None
povGroupMember	The POV group member for which to clear model artifacts and data, such as 2015_January_Actual	Path	Yes	None
isManageRule	To clear the program rule details or not; true/false	Payload	No	None
isInputData	To clear input data or not; true/false	Payload	No	None
queryName	A query name already existing within the application; used to clear a region within the given POV	Payload	No	None
isAllocatedValues	To clear allocation values or not; true/false	Payload	No	None
isAdjustmentValues	To clear adjustment values or not; true/false	Payload	No	None
stringDelimiter	String delimiter for POV group members	Payload	No	"_" (Underscore)

 **Note:**

If `queryName` is used (is not null), then `isManageRule`, `isAllocatedValues`, and `isAdjustmentValues` must be set to `false`.

If one of these parameters or `isInputData` is not passed, it is considered as `false`.

Example URL and payload to clear to a particular region within input data

```
https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/
rest/{api_version}/applications/{application}/povs/{povGroupMember}/jobs/
clearPOVJob
```

```
{"isInputData":"true","queryName":"myQueryName","stringDelimiter":"_"}
```

Response Body

Supported Media Types: `application/json`

Table 16-36 Parameters

Name	Description
details	Task ID, such as BksML1_BksML1_ClearMLPOV_D20160113T0703 58_1da_1
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type":"Profitability",
  "status":-1,
  "statusMessage":"In Progress",
  "details":"BksML30_ClearMLPOV_D20220511T114821_f4b",
  "links":[
    {
      "href":"http:// :// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/applications/
jobs/ChecktaskStatusJob/BksML30_ClearMLPOV_D20220511T114821_f4b",
      "action":"GET",
      "rel":"Job Status"
    }
  ]
}
```

```

    }
  ]
}

```

cURL Sample – ClearPOV.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

uncClearPOVData() {
    stringDelimiter="_";

    param="{\"isManageRule\": \"true\", \"isInputData\": \"true\", \"stringDelimit
    er\": \"${stringDelimiter}\""
        url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/
    povs/$POV_GROUP_MEMBER/jobs/clearPOVJob
        funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started Clearing POV successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – ClearPOV.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def clearPOVData() {

    JSONObject json = new JSONObject();
    json.put("isManageRule", true);
    json.put("isInputData", true);
    json.put("stringDelimiter", "_");

    String povGroupMember = "2014_January_Actual";

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/
    applications/" + appName + "/povs/"
        +
    povGroupMember.trim().replaceAll(" ", "%20") + "/jobs/clearPOVJob";
}

```

```
def url;

try {
    url = new URL(urlString)
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}

executeJob(url, "POST", json.toString());
}
```

Java Sample – clearPOV.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void clearPOVData() throws Exception {

    JSONObject json = new JSONObject();
    json.put("isManageRule", true);
    json.put("isInputData", true);
    json.put("stringDelimiter", "_");

    String povGroupMember = "2014_January_Actual";

    String urlString = "%s/epm/rest/%s/applications/%s/povs/" +
        povGroupMember.trim().replaceAll(" ", "%20")
        + "/jobs/clearPOVJob";
    executeJob(urlString, "POST", json.toString());
}
```

Run ML Rule Balancing

Retrieves Rule Balancing data for a particular POV for a given application.

Required Roles

Service Administrator, Power User

REST Resource

GET /epm/rest/{api_version}/applications/{application}/povs/{povGroupMember}/ruleBalance?queryParameter={"modelName":"modelName"}

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-37 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application for which to retrieve rule balancing data	Path	Yes	None
povGroupMember	POV name for which to retrieve the results, such as 2015_January_Actual	Path	Yes	None
modelViewName	Model view name to filter the results within the POV area	Query	Yes	None
stringDelimiter	String delimiter for POV group members, such as "_"	Query	No	Underscore, "_"

Example URL and Sample Query Parameter

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/{api_version}/
applications/{application}/povs/{povGroupMembers}/ruleBalance?
queryParameter={"modelViewName":"modelViewName"}
```

Response Body

Supported Media Types: application/json

Table 16-38 Parameters

Name	Description
details	Rule balancing output for the given POV
status	See Migration Status Codes
statusMessage	Message about the status, such as Success
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
"items": [{
  "ruleNumber": "",
  "rules": [],
  "balanceTypeRule": true,
  "scale": 2,
  "sequence": 0,
  "name": "NoRule",
```

```

        "description": null,
        "runningBalance": 49357098.03,
        "balance": 49357098.03,
        "allocationIn": null,
        "allocationOut": null,
        "adjustmentIn": null,
        "adjustmentOut": null,
        "input": 49357098.03,
        "runningRemainder": 49357098.03,
        "remainder": 49357098.03,
        "netChange": null,
        "offset": null,
        "inputAsString": "49,357,098.03",
        "adjInAsString": "-",
        "adjOutAsString": "-",
        "allocInAsString": "-",
        "allocOutAsString": "-",
        "balanceAsString": "49,357,098.03",
        "runningBalanceAsString": "49,357,098.03",
        "runningRemainderAsString": "49,357,098.03",
        "remainderAsString": "49,357,098.03",
        "netChangeAsString": "-",
        "offsetAsString": "-"
    },
    ],
    "type": "Profitability",
    "status": 0,
    "details": "",
    "statusMessage": "Success"
}

```

Java Sample – RunRuleBalancing.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void runRuleBalancing() throws Exception {

    String modelViewName = null;

    JSONObject json = new JSONObject();
    json.put("stringDelimiter", "_");
    json.put("modelViewName", modelViewName);

    String povGroupMember = "2014_January_Actual";

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
        applications/" + applicationName + "/povs/"
        + povGroupMember.trim().replaceAll("
", "%20") + "/ruleBalance";
    urlString = urlString + "?" + "queryParameter=" + json.toString();
}

```

```

        String response = executeRequest(urlString, "GET", null,
"application/json");
        JSONObject jsonObj = new JSONObject(response);
        int resStatus = jsonObj.getInt("status");

        if(resStatus == 0) {
            System.out.println("Rule Balancing ran successfully");
            JSONArray itemsArray =
jsonObj.getJSONArray("items");
            System.out.println("Details : " + itemsArray.toString());
        } else {
            String details = jsonObj.getString("details");
            System.out.println("Rule Balancing failed. Details : " +
details);
        }
    }
}

```

cURL Sample – RunRuleBalancing.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcRunRuleBalancing() {
    url=$SERVER_URL/epm/rest/$API_VERSION/applications/$APP_NAME/
povs/$POV_GROUP_MEMBER/ruleBalance
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"
    list=`cat response.txt | jq 'select(.items != null)
| .items[].name'`
    if [[ ! -z $list ]]; then
        echo $list
    else
        echo "No Items found"
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

Groovy Sample – RunRuleBalancing.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def runRuleBalancing() {

    String modelAndViewName = null;

```

```

JSONObject json = new JSONObject();
json.put("stringDelimiter", "_");
json.put("modelName", modelName);

String povGroupMember = "2014_January_Actual";

def url;
def response;

String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
applications/" + appName + "/povs/"
                                + povGroupMember.trim().replaceAll("
", "%20") + "/ruleBalance";
urlString = urlString + "?" + "queryParameter=" + json.toString();

try {
    url = new URL(urlString);
} catch (MalformedURLException e) {
    println "Malformed URL. Please pass valid URL"
    System.exit(0);
}

response = executeRequest(url, "GET", null, "application/json");
JSONObject jsonObj = new JSONObject(response);
int resStatus = jsonObj.getInt("status");

if(resStatus == 0) {
    println "Rule Balancing ran successfully"
    JSONArray itemsArray = jsonObj.getJSONArray("items");
    println "Details : " + itemsArray.toString()
} else {
    String details = jsonObj.getString("details");
    println "Rule Balancing failed. Details : " + details
}
}

```

Update File-Based Application

Uploads a new dimension flat file for an application created using a flat file.

Required Roles

Service Administrator, Power User

REST Resource

POST/epm/rest/{api_version}/fileApplications/{application}/updateDimension

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-39 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application to update	Path	Yes	None
dataFileName	Dimension Metadata flat file name that has already been uploaded to the Inbox folder	Payload	Yes	None

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcx>.oraclecloud.com/epm/rest/v1/
fileApplications/BksML12/jobs/updateDimension{"dataFileName":"input.txt"}
```

Response Body

Supported Media Types: application/json

Table 16-40 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_UpdateDimension_D20160118T051020_bb8_1
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type":"Profitability",
  "status":0,
  "statusMessage":"Success",
  "details":"true"
}
```

Java Sample – UpdateDimension.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```
public void updateDimensions() throws Exception {

    JSONObject json = new JSONObject();
    json.put("dataFileName", "Accounts.txt");

    String urlString = "%s/epm/rest/%s/fileApplications/%s/
updateDimension";
    String response = executeRequest(urlString, "POST", json.toString(),
"application/json");
    JSONObject jsonObj = new JSONObject(response);
    int resStatus = jsonObj.getInt("status");

    if(resStatus == 0) {
        System.out.println("Dimensions updated successfully");
    } else {
        System.out.println("Dimensions update failed");
    }
}
```

cURL Sample – UpdateDimension.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```
funcUpdateDimensions() {
    dataFileName="Accounts.txt"
    param="{\"dataFileName\": \"${dataFileName}\"}"
    url=$SERVER_URL/epm/rest/$API_VERSION/fileApplications/$APP_NAME/
updateDimension
    funcExecuteRequest "POST" $url "$param" "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Dimensions updated successfully"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}
```

Groovy Sample – UpdateDimension.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```
def updateDimensions() {

    JSONObject json = new JSONObject();
    json.put("dataFileName", "Accounts.txt");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
fileApplications/" + appName + "/updateDimension";
    def url;

    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }

    String response = executeRequest(url, "POST", json.toString(),
"application/json");
    JSONObject jsonObj = new JSONObject(response);
    int resStatus = jsonObj.getInt("status");

    if(resStatus == 0) {
        println "Dimensions updated successfully"
    } else {
        println "Dimensions update failed"
    }
}
```

Update Dimensions As a Job

Uploads a new dimension flat file for an application created using a flat file.

Similar to [Update File-Based Application](#), you can use the new resource to update dimension flat files for an application. However, Update Dimensions As a Job runs asynchronously; it immediately returns the job ID and the job status (Running or Failed). Update File-Based Application runs synchronously and waits until the job finishes to indicate whether the job succeeded or failed.

Required Roles

Service Administrator, Power User

REST Resource

POST/epm/rest/{api_version}/fileApplications/{application}/jobs/
updateDimension

Request

Supported Media Types: application/json

The following table summarizes the client request.

Table 16-41 Parameters

Name	Description	Type	Required	Default
api_version	Version of the API you are developing with	Path	Yes	None
application	Name of the application to update	Path	Yes	None
dataFileName	Dimension Metadata flat file name that has already been uploaded to the Inbox folder; multiple file names can be passed separated by comma or other separator character listed in the stringDelimiter parameter	Payload	Yes	None
stringDelimiter	Separator character to use if different from commas	Payload	No	Comma (,)

Example URL and Payload

```
https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/<api_version>/
fileApplications/BksML12/jobs/
updateDimension{"dataFileName":"input.txt","stringDelimiter":","}
```

Response Body

Supported Media Types: application/json

Table 16-42 Parameters

Name	Description
details	Task ID, such as BksML12_BksML12_UpdateDimension_D20160118T051020_bb8_1
status	See Migration Status Codes
statusMessage	Message about the status, such as In Progress
type	Profitability
data	Parameters as key value pairs
links	Detailed information about the link
href	Links to API call
action	The HTTP call type
rel	Relationship type
data	Parameters as key value pairs passed in the request

Example of Response Body

The following shows an example of the response body in JSON format.

```
{
  "type":"Profitability",
  "status":-1,
  "statusMessage":"In Progress",
  "details":"BksML30_UpdateDimensions_D20220513T062046_c61",
  "links":[
    {
```

```

        "href":"http:// :// <SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/epm/rest/v1/
applications/jobs/ChecktaskStatusJob/
BksML30_UpdateDimensions_D20220513T062046_c61",
        "action":"GET",
        "rel":"Job Status"
    }
}
]
}

```

Java Sample – UpdateDimensionJob.java for Profitability and Cost Management

Prerequisites: json.jar

Prerequisites: See [Profitability and Cost Management Common Helper Functions for Java](#)

```

public void updateDimensionJob() throws Exception {

    JSONObject json = new JSONObject();
    json.put("dataFileName", "Accounts.txt,Activity.txt");

    String urlString = serverURL + "/epm/rest/" + apiVersion + "/"
fileApplications/" + applicationName + "/updateDimensionJob";

    exe4cuteJob(urlString, "POST", json.toString());
}

```



Note:

In the main method, enter the following statement:

```
restSamplesObj.updateDimensionsJob();
```

cURL Sample – UpdateDimensionJob.sh for Profitability and Cost Management

Prerequisites: jq <http://stedolan.github.io/jq/download/linux64/jq>

Common functions: See [Profitability and Cost Management Common Helper Functions for cURL](#).

```

funcUpdateDimensionJob() {
    dataFileName="Accounts.txt,Activity.txt"
    param="{\"dataFileName\": \"\$dataFileName\"}"
    url=\$SERVER_URL/epm/rest/\$API_VERSION/fileApplications/\$APP_NAME/
updateDimensionJob
    funcExecuteRequest "POST" \$url "\$param" "application/json"
}

```

```

output=`cat response.txt`
status=`echo $output | jq '.status'`
if [ $status == -1 ]; then
    echo "Started Update Dimensions Job successfully"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred." $error
fi
    funcRemoveTempFiles "respHeader.txt" response.txt"
}

```

**Note:**

At the end, call this statement along with other statements:

```
funcUpdateDimensionsJob
```

Groovy Sample – UpdateDimensionJob.groovy for Profitability and Cost Management

Prerequisites: json.jar

Common functions: See [Appendix C: Common Helper Functions for Groovy](#).

```

def updateDimensionsJob() {

    JSONObject json = new JSONObject();
    json.put("dataFileName", "Accounts.txt,Activity.txt");

    String urlString = serverUrl + "/epm/rest/" + apiVersion + "/"
fileApplications/" + appName + "/updateDimensionJob";
    def url;

    try {
        url = new URL(urlString)
    } catch (MalformedURLException e) {
        println "Malformed URL. Please pass valid URL"
        System.exit(0);
    }

    executeJob(url,"POST",json.toString());
}

```

 **Note:**

In the main method, add the following statement:

```
restSamplesObj.updateDimensionJob();
```

17

Narrative Reporting REST APIs

You can use the [REST APIs for Narrative Reporting](#) to work with Narrative Reporting artifacts, report packages, report snapshots, and reports.

You can use the [REST APIs for Narrative Reporting](#) to execute these actions:

- **Files**
 - Download a file from the temporary repository or to the Library.
 - Upload a temporary file to the Narrative Reporting repository.
- **Jobs**
 - Start a new Narrative Reporting job for asynchronous execution by the service.
 - Get a Narrative Reporting job's status that provides links to the job results when the job is complete.
- **Reports**
 - You can get the corresponding Report information
 - You can get the global point of view selections and member suggestions for the Report
 - You can get the required member selection prompts for the Report
- **Books**
 - You can get the corresponding Book information
 - You can get the global point of view selections and member suggestions for Book
- **Bursting Definitions**
 - You can view the Bursting Definitions and their content
 - You can execute the Bursting Definition file

18

Enterprise Data Management Cloud REST APIs

Use the [REST APIs for Enterprise Data Management Cloud](#) to work with applications, files, jobs, requests, and views.

A

Common Helper Functions for Java

This appendix shows the common helper functions for Java for the EPM REST APIs.

Note: The `userName` variable uses the format `<domain>.<username>`. See [Authentication](#).

```
/*
   File: PbcRestSamples.java - Created on Feb 19, 2015
   Copyright (c) 2015 Oracle Corporation. All Rights Reserved.
   This software is the proprietary information of Oracle.
*/
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Scanner;

import org.json.JSONArray;
import org.json.JSONObject;

/*
 * PBCS Rest Samples.
 * The userName variable uses the format <domain>.<username>.
 */
public class PbcRestSamples{
    private String userName;           // PBCS user name
    private String password;          // PBCS user password
    private String serverUrl;         // PBCS server URL
    private String apiVersion;        // Version of the PBCS API that you
are developing/compiling with.
    private String applicationName;    // PBCS application used in this sample

    public static void main(String[] args) {
        try {
            PbcRestSamples samples = new
PbcRestSamples("epm_default_cloud_admin", "epm_cloud", "https://
<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/",
"11.1.2.3.600", "Vision");
            samples.integrationScenarioImportMetadataIntoApplication();
            samples.integrationScenarioImportDataRunCalcCopyToAso();
        }
    }
}
```

```

samples.integrationScenarioExportMetadataAndDataAndDownloadFiles();
    samples.integrationScenarioRemoveUnnecessaryFiles();

samples.integrationScenarioExportDataAndDownloadFiles();
    samples.integrationScenarioRefreshTheApplication();
} catch (Throwable x) {
    System.err.println("Error: " + x.getMessage());
}
}

    public PbcRestSamples(String userName, String password, String
serverUrl, String apiVersion, String applicationName) throws Exception
{
    this.userName = userName;
    this.password = password;
    this.serverUrl = serverUrl;
    this.apiVersion = apiVersion;
    this.applicationName = applicationName;
}

//
// BEGIN - Integration scenarios.
//
    public void integrationScenarioImportMetadataIntoApplication()
throws Exception {
        uploadFile("accounts.zip");
        executeJob("IMPORT_METADATA", "accountMetadata",
"{importZipFileName:accounts.zip}");
        executeJob("CUBE_REFRESH", null, null);
    }

    public void integrationScenarioImportDataRunCalcCopyToAso() throws
Exception {
        uploadFile("data.csv");
        executeJob("IMPORT_DATA", "loadingqldata",
"{importFileName:data.csv}");
        executeJob("CUBE_REFRESH", null, null);
        executeJob("PLAN_TYPE_MAP", "CampaignToReporting",
"{clearData:false}");
    }

    public void
integrationScenarioExportMetadataAndDataAndDownloadFiles() throws
Exception {
        executeJob("EXPORT_METADATA", "exportentitymetadata",
"{exportZipFileName:entitydata.zip}");
        executeJob("EXPORT_DATA", "Forecastdata",
"{exportFileName:forecastdata.zip}");
        listFiles();
        downloadFile("entitydata.zip");
        downloadFile("forecastdata.zip");
    }

    public void integrationScenarioRemoveUnnecessaryFiles() throws
Exception {

```

```

        listFiles();
        deleteFile("entitymetadata.csv");
        deleteFile("forecastdata.csv");
    }

    public void integrationScenarioExportDataAndDownloadFiles() throws
Exception {
        executeJob("EXPORT_DATA", "entitydata",
"{exportFileName:entitydata.zip}");
        executeJob("EXPORT_DATA", "forecastdata",
"{exportFileName:forecastdata.zip}");
        listFiles();
        downloadFile("entitydata.zip");
        downloadFile("forecastdata.zip");
    }

    public void integrationScenarioRefreshTheApplication() throws Exception {
        uploadFile("accounts.zip");
        executeJob("IMPORT_METADATA", "accountMetadata",
"{importZipFileName:accounts.zip}");
        executeJob("CUBE_REFRESH", null, null);
    }

    public void integrationScenarioCloneServiceInstance() throws Exception {
        // Part 1 : Change serverUrl, username, password, apiVersion
variables values to match those of first environment
        // Download file from source instance.
        // Comment out all lines below Part 2
        // Uncomment the below line for the first step.
        // downloadFile("Artifact Snapshot");

        // Part 2 : Change serverUrl, username, password, apiVersion to
match those of second environment.
        // Clone the service instance.
        // Comment out code for download file.
        // Uncomment below lines
        recreateService("PBCS");
        deleteFile("Artifact Snapshot");
        uploadFile("Artifact Snapshot.zip");
        importSnapshot("Artifact Snapshot");
    }
    //
    // END - Integration scenarios.
    //

    //
    // BEGIN - Methods that invoke REST API
    //

    //
    // Common Helper Methods
    //
    private String getStringFromInputStream(InputStream is) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();

```

```
String line;

try {
    br = new BufferedReader(new InputStreamReader(is));
    while ((line = br.readLine()) != null) {
        sb.append(line);
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (br != null) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

return sb.toString();
}

private String executeRequest(String urlString, String
requestMethod, String payload, String contentType) throws Exception {
    HttpURLConnection connection = null;
    try {
        URL url = new URL(urlString);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod(requestMethod);
        connection.setInstanceFollowRedirects(false);
        connection.setDoOutput(true);
        connection.setUseCaches(false);
        connection.setDoInput(true);
        connection.setRequestProperty("Authorization", "Basic " +
new sun.misc.BASE64Encoder().encode((userName + ":" +
password).getBytes()));
        connection.setRequestProperty("Content-Type", contentType);
        if (payload != null) {
            OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
            writer.write(payload);
            writer.flush();
        }
        int status = connection.getResponseCode();
        if (status == 200 || status == 201) {
            return
getStringFromInputStream(connection.getInputStream());
        }
        throw new Exception("Http status code: " + status);
    } finally {
        if (connection != null)
            connection.disconnect();
    }
}
}
```

```

        private void getJobStatus(String pingUrlString, String methodType)
        throws Exception {
            boolean completed = false;
            while (!completed) {
                String pingResponse = executeRequest(pingUrlString, methodType,
                null, "application/x-www-form-urlencoded");
                JSONObject json = new JSONObject(pingResponse);
                int status = json.getInt("status");
                if (status == -1) {
                    try {
                        System.out.println("Please wait...");
                        Thread.sleep(20000);
                    } catch (InterruptedException e) {
                        completed = true;
                        throw e;
                    }
                }
                else {
                    if (status > 0) {
                        System.out.println("Error occurred: " +
                json.getString("details"));
                    }
                    else {
                        System.out.println("Completed");
                    }
                    completed = true;
                }
            }
        }
    }

```

```

        private void getMigrationJobStatus(String pingUrlString, String
        methodType) throws Exception {
            boolean completed = false;
            while (!completed) {
                String pingResponse = executeRequest(pingUrlString, methodType,
                null, "application/x-www-form-urlencoded");
                JSONObject json = new JSONObject(pingResponse);
                int status = json.getInt("status");
                if (status == -1) {
                    try {
                        System.out.println("Please wait...");
                        Thread.sleep(20000);
                    } catch (InterruptedException e) {
                        completed = true;
                        throw e;
                    }
                }
                else {
                    if (status == 1){
                        System.out.println("Error occured");
                        JSONArray itemsArray =
                json.getJSONArray("items");
                        JSONObject jObj = null;
                        if(itemsArray.length() <= 0){
                            System.out.println(json.getString("details"));
                        }
                    }
                }
            }
        }
    }

```

```

    }else{
        for (int i=0; i < itemsArray.length(); i++){
            jsonObj = (JSONObject)itemsArray.get(i);
            String source = jsonObj.getString("source");
            String destination =
jObj.getString("destination");
            String taskURL = null;
            JSONArray lArray = jsonObj.getJSONArray("links");
            for (int j = 0; j < lArray.length(); j++) {
                JSONObject arr = lArray.getJSONObject(j);
                if (!JSONObject.NULL.equals(arr) && !
JSONObject.NULL.equals(arr.get("rel")) && arr.get("rel").equals("Job
Details")) {
                    taskURL = (String) arr.get("href");
                    break;
                }
            }
            System.out.println("Details:");
            System.out.println("Source: " + source);
            System.out.println("Destination: "+
destination);

            boolean errorsCompleted = false;
            String currentMessageCategory = "";
            String nextPingURL = taskURL;
            while(!errorsCompleted){
                String nextPingResponse =
executeRequest(nextPingURL, "GET", null, "application/x-www-form-
urlencoded");
                JSONObject jsonObj = new
JSONObject(nextPingResponse);
                int status1 = jsonObj.getInt("status");
                if(status1 == 0){
                    JSONArray artifactArray =
jsonObj.getJSONArray("items");
                    JSONObject jRes = null;
                    for(int k=0; k <
artifactArray.length(); k++){
                        jRes =
(JSONObject)artifactArray.get(k);
                        String artifact =
jRes.getString("artifact").toString();
                        String msgCategory =
jRes.getString("msgCategory").toString();
                        String msgText =
jRes.getString("msgText").toString();

                        if(currentMessageCategory.isEmpty() || !
currentMessageCategory.equals(msgCategory)){
                            currentMessageCategory =
msgCategory;

                            System.out.println(currentMessageCategory);
                        }
                        System.out.println(artifact + " - "
+ msgText);

```



```

// BEGIN - List all the versions in PBCS
//
public void getLCMVersions() throws Exception {
    String urlString = String.format("%s/interop/rest", serverUrl);
    String response = executeRequest(urlString, "GET", null,
"application/x-www-form-urlencoded");
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        JSONArray fileList = json.getJSONArray("items");
        System.out.println("List of files are :");
        JSONObject jsonObj = null;
        for(int i=0; i<fileList.length(); i++){0) {
            jsonObj = (JSONObject)fileList.get(i);
            System.out.println("Version :" +
jsonObj.getString("version"));
            System.out.println("Lifecycle :" +
jsonObj.getString("lifecycle"));
            System.out.println("Latest :" +
jsonObj.getString("latest"));
            System.out.println("Link :" + ((JSONObject)
((JSONArray) jsonObj.getJSONArray("links")).get(0)).getString("href") +
"\n");
        }
    }
}
//
// END - List all the versions in PBCS
//

//
// BEGIN - Get application snapshot details
//
public void getApplicationSnapshotDetails(String snapshotName)
throws Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots/%s", serverUrl, apiVersion, snapshotName);
    String response = executeRequest(urlString, "GET", null,
"application/x-www-form-urlencoded");
    JSONObject json = new JSONObject(response);

    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        System.out.println("Application details :");
        JSONArray itemsArray = json.getJSONArray("items");
        JSONObject item = (JSONObject) itemsArray.get(0);
        System.out.println("Application snapshot name : " +
item.getString("name"));
        System.out.println("Application snapshot type : " +
item.getString("type"));
        System.out.println("Can be exported flag : " +
item.getString("canExport"));
        System.out.println("Can be imported flag : " +
item.getString("canImport"));
        System.out.println("Can be uploaded flag : " +

```



```

item.getString("canUpload"));
        System.out.println("Can be downloaded flag : " +
item.getString("canDownload"));

        JSONArray linksArray = json.getJSONArray("links");
        JSONObject jsonObj = null;
        System.out.println("Services details :");
        for(int i=0; i < linksArray.length(); i++){
            jsonObj = (JSONObject)linksArray.get(i);
            System.out.println("Service :" + jsonObj.getString("rel"));
            System.out.println("URL :" + jsonObj.getString("href"));
            System.out.println("Action :" + jsonObj.getString("action") +
"\n");
        }
    }
}
//
// END - Get application snapshot details
//

//
// BEGIN - List all the files in PBCS
//
public void listFiles() throws Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots", serverUrl, apiVersion);
    String response = executeRequest(urlString, "GET", null,
"application/x-www-form-urlencoded");
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        if (json.get("items").equals(JSONObject.NULL))
            System.out.println("No files found");
        else {
            System.out.println("List of files :");
            JSONArray itemsArray =
json.getJSONArray("items");
            JSONObject jsonObj = null;
            for (int i=0; i < itemsArray.length(); i++){
                jsonObj = (JSONObject)itemsArray.get(i);
                System.out.println(jsonObj.getString("name"));
            }
        }
    }
}
//
// END - List all the files in PBCS
//

//
// BEGIN - Delete a file in PBCS
//
public void deleteFile(String fileName) throws Exception {
    String urlString = String.format("%s/interop/rest/%s/
applicationsnapshots/%s", serverUrl, apiVersion, fileName);

```

```

        String response = executeRequest(urlString, "DELETE", null,
"application/x-www-form-urlencoded");
        JSONObject json = new JSONObject(response);
        int resStatus = json.getInt("status");
        if (resStatus == 0)
            System.out.println("File deleted successfully");
        else
            System.out.println("Error deleting file : " +
json.getString("details"));
    }
    //
    // END - Delete a file in PBCS
    //

    //
    // BEGIN - Download a file from PBCS
    //
    public void downloadFile(String fileName) throws Exception {
        HttpURLConnection connection = null;
        InputStream inputStream = null;
        FileOutputStream outputStream = null;

        try {
            URL url = new URL(String.format("%s/interop/rest/%s/
applicationsnapshots/%s/contents", serverUrl, apiVersion, fileName));
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setInstanceFollowRedirects(false);
            connection.setDoOutput(true);
            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setRequestProperty("Authorization", "Basic " +
new sun.misc.BASE64Encoder().encode((userName + ":" +
password).getBytes()));
            connection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
            int status = connection.getResponseCode();
            if (status == 200) {
                if (connection.getContentType() != null &&
connection.getContentType().equals("application/json")) {
                    JSONObject json = new
JSONObject(getStringFromInputStream(connection.getInputStream()));
                    System.out.println("Error downloading file : " +
json.getString("details"));
                } else {
                    inputStream = connection.getInputStream();
                    outputStream = new FileOutputStream(new
File(fileName));

                    int bytesRead = -1;
                    byte[] buffer = new byte[5 * 1024 * 1024];
                    while ((bytesRead = inputStream.read(buffer)) !=
-1)

                        outputStream.write(buffer, 0, bytesRead);
                    System.out.println("File download completed.");
                }
            }
        }
    }

```

```

        } else {
            throw new Exception("Http status code: " + status);
        }
    } finally {
        if (connection != null)
            connection.disconnect();
        if (outputStream != null)
            outputStream.close();
        if (inputStream != null)
            inputStream.close();
    }
}
//
// END - Download a file from PBCS
//

//
// BEGIN - Upload a file to PBCS
//
public void uploadFile(String fileName) throws Exception {
    final int DEFAULT_CHUNK_SIZE = 50 * 1024 * 1024;
    InputStream fis = null;
    byte[] lastChunk = null;
    long totalFileSize = new File(fileName).length(), totalbytesRead = 0;
    boolean isLast = false, status = true;
    Boolean isFirst = true;
    int packetNo = 1, lastPacketNo = (int) (Math.ceil(totalFileSize /
(double) DEFAULT_CHUNK_SIZE));

    try {
        fis = new BufferedInputStream(new FileInputStream(fileName));
        while (totalbytesRead < totalFileSize && status) {
            int nextChunkSize = (int) Math.min(DEFAULT_CHUNK_SIZE,
totalFileSize - totalbytesRead);
            if (lastChunk == null) {
                lastChunk = new byte[nextChunkSize];
                totalbytesRead += fis.read(lastChunk);
                if (packetNo == lastPacketNo)
                    isLast = true;
                status = sendFileContents(isFirst, isLast, lastChunk,
fileName);

                isFirst=false;
                packetNo = packetNo + 1;
                lastChunk = null;
            }
        }
        System.out.println("Uploaded successfully");
    } finally {
        if (fis != null)
            fis.close();
    }
}

private boolean sendFileContents(Boolean isFirst, boolean isLast, byte[]
lastChunk, String fileName) throws Exception {

```

```

        HttpURLConnection connection = null;

        try {
            URL url = new URL(String.format("%s/interop/rest/%s/
applicationsnapshots/%s/contents?
q={chunkSize:%d,isFirst:%b,isLast:%b}",
                serverUrl, apiVersion, fileName, lastChunk.length,
isFirst, isLast));
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setInstanceFollowRedirects(false);
            connection.setDoOutput(true);
            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setRequestProperty("Authorization", "Basic " +
new sun.misc.BASE64Encoder().encode((userName + ":" +
password).getBytes()));
            connection.setRequestProperty("Content-Type", "application/
octet-stream");

            DataOutputStream wr = new
DataOutputStream(connection.getOutputStream());
            wr.write(lastChunk);
            wr.flush();

            int statusCode = connection.getResponseCode();
            String status =
getStringFromInputStream(connection.getInputStream());
            if (statusCode == 200 && status != null) {
                int commandStatus = getCommandStatus(status);
                if (commandStatus == 0) {
                    isFirst = false;
                    return true;
                } else if (commandStatus == -1 && isLast) {
                    getJobStatus(fetchPingUrlFromResponse(status, "Job
Status"), "GET");
                }
            }

            return false;
        } finally {
            if (connection != null)
                connection.disconnect();
        }
    }

    public int getCommandStatus(String response) throws Exception {
        JSONObject json = new JSONObject(response);
        if (!JSONObject.NULL.equals(json.get("status")))
            return json.getInt("status");
        else
            return Integer.MIN_VALUE;
    }
    //
    // END - Upload a file to PBCS

```

```
//  
  
//  
// BEGIN - Import an application snapshot  
//  
public void importSnapshot(String applicationSnapshotName) throws  
Exception {  
    JSONObject params = new JSONObject();  
    params.put("type", "import");  
    String urlString = String.format("%s/interop/rest/%s/  
applicationsnapshots/%s/migration?q=%s", serverUrl, apiVersion,  
applicationSnapshotName, params.toString());  
    String response = executeRequest(urlString, "POST", null,  
"application/x-www-form-urlencoded");  
    System.out.println("Import started successfully");  
    getMigrationJobStatus(fetchPingUrlFromResponse(response, "Job  
Status"), "POST");  
}  
//  
// END - Import an application snapshot  
//  
  
//  
// BEGIN - Export an application snapshot  
//  
public void exportSnapshot(String applicationSnapshotName) throws  
Exception {  
    JSONObject params = new JSONObject();  
    params.put("type", "export");  
    String urlString = String.format("%s/interop/rest/%s/  
applicationsnapshots/%s/migration?q=%s", serverUrl, apiVersion,  
applicationSnapshotName, params.toString());  
    String response = executeRequest(urlString, "POST", null,  
"application/x-www-form-urlencoded");  
    System.out.println("Export started successfully");  
    getMigrationJobStatus(fetchPingUrlFromResponse(response, "Job  
Status"), "POST");  
}  
//  
// END - Export an application snapshot  
//  
  
//  
// BEGIN - Provide Feedback  
//  
public void provideFeedback(String description) throws Exception {  
    JSONObject params = new JSONObject();  
    JSONObject config = new JSONObject();  
    config.put("URL", serverUrl);  
    params.put("configuration", config);  
    params.put("description", description);  
  
    String urlString = String.format("%s/interop/rest/%s/feedback",  
serverUrl, apiVersion);  
    String response = executeRequest(urlString, "POST",
```

```

params.toString(), "application/json");
    JSONObject json = new JSONObject(response);
    int resStatus = json.getInt("status");
    if (resStatus == 0) {
        System.out.println("Feedback successful");
    } else {
        System.out.println("Error occured: " +
json.getString("details"));
    }
}
//
// END - Provide Feedback
//

//
// BEGIN - Reset services
//
public void hardReset(String comment) throws Exception {
    Scanner in = new Scanner(System.in);
    System.out.println("Are you sure you want to restart the
service instance (yes/no): no ?[Press Enter]");
    String s = in.nextLine();
    if (!s.equals("yes")) {
        System.out.println("User cancelled the recreate command");
        System.exit(0);
    }

    JSONObject params = new JSONObject();
    params.put("comment", java.net.URLEncoder.encode(comment));

    String urlString = String.format("%s/interop/rest/%s/services/
PBCS/resetservice", serverUrl, apiVersion);
    String response = executeRequest(urlString, "POST",
params.toString(), "application/x-www-form-urlencoded");
    waitForCompletion(fetchPingUrlFromResponse(response, "Job
Status"));
}
//
// END - Reset services
//
//
// BEGIN - Execute a Job (EXPORT_DATA, EXPORT_METADATA,
IMPORT_DATA, IMPORT_METADATA, CUBE_REFRESH, ...)
//
public void executeJob(String jobType, String jobName, String
parameters) throws Exception {
    String urlString = String.format("%s/HyperionPlanning/rest/%s/
applications/%s/jobs", serverUrl, apiVersion, applicationName);
    JSONObject payload = new JSONObject();
    payload.put("jobName", jobName);
    payload.put("jobType", jobType);
    payload.put("parameters", new JSONObject(parameters));
    String response = executeRequest(urlString, "POST",
payload.toString(), "application/json");
    System.out.println("Job started successfully");
}

```

```
        getJobStatus(fetchPingUrlFromResponse(response, "self"),
"GET");
    }
    //
    // END - Execute a Job (EXPORT_DATA, EXPORT_METADATA, IMPORT_DATA,
IMPORT_METADATA, CUBE_REFRESH, ...)
    //
}
```

 **Note:**

Note on Proxy Setting: In case of proxies, set the proxy host and port as the system arguments.

B

CSS Common Helper Functions for Java

This appendix shows the CSS common helper functions for Java for the EPM REST APIs.

Prerequisites: json.jar

Note on Proxy Setting: In case of proxies, set the proxy host and port as the system arguments.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import javax.xml.bind.DatatypeConverter;

import org.json.JSONArray;
import org.json.JSONObject;

public class CSSRESTSamples {
    private static String userName;
    private static String password;
    private String serverUrl;
    private String apiVersion;

    public static void main(String[] args) {
        try {
            CSSRESTSamples samples = new CSSRESTSamples("<DOMAINNAME.USERNAME>",
"<PASSWORD>", "<SERVICE_URL>", "v1");
            // Call sample APIs.
            // samples.addUsers("AddUser2.csv", "test123$", false);
            // samples.removeUsers("test2.csv");
            // samples.assignRole("test3.csv", "Power User");
            // samples.unassignRole("test4.csv", "Viewer");
            // samples.addUsersToGroup("test5.csv", "TestGroup1");
            // samples.removeUsersFromGroup("test6.csv", "TestGroup2");
            // samples.generateRoleAssignmentReport("JavaSampleReport.csv");
            // samples.generateUserGroupReport("UserGroupReport.csv");
            // samples.addUserToGroups("Group.csv", "user1");
            // samples.removeUserFromGroups("groups.csv", "joe");
            // samples.addGroups("Group1.csv");
        }
    }
}
```



```

        // samples.removeGroups("DeleteGroup1.csv");
        // samples.generateInvalidLoginReport("2021-06-01",
"2021-06-10", "invalidLoginReport141.csv");
        // samples.generateRoleAssignmentAuditReport("2021-06-01",
"2021-06-10", "roleAssignmentaudit_14778.csv");
        // samples.updateUsers("updateuser.csv");
    } catch (Throwable x) {
        System.err.println("Error: " + x.getMessage());
    }
}

    public CSSRESTSamples(String userName, String password, String
serverUrl, String apiVersion) throws Exception {
        this.userName = userName;
        this.password = password;
        this.serverUrl = serverUrl;
        this.apiVersion = apiVersion;
    }

    public void addUsers(String fileName, String userPassword, boolean
resetPassword) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
            Map<String, String> reqHeaders = new HashMap<String,
String>();
            reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
                .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
String>();
            reqParams.put("filename", fileName);
            reqParams.put("userpassword", userPassword);
            reqParams.put("resetpassword", resetPassword + "");

            Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                "POST");
            String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void removeUsers(String fileName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
            Map<String, String> reqHeaders = new HashMap<String,
String>();
            reqHeaders.put("Authorization", "Basic " +

```

```

DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset()));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "DELETE");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void assignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ASSIGN_ROLE");
        reqParams.put("rolename", roleName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void unassignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "UNASSIGN_ROLE");

```

```

        reqParams.put("rolename", roleName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void addUsersToGroup(String fileName, String groupName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ADD_USERS_TO_GROUP");
        reqParams.put("groupname", groupName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void removeUsersFromGroup(String fileName, String
groupName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();

```

```

        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "REMOVE_USERS_FROM_GROUP");
        reqParams.put("groupname", groupName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void addUserToGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ADD_USER_TO_GROUPS");
        reqParams.put("username", userName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void removeUserFromGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "REMOVE_USER_FROM_GROUPS");
        reqParams.put("username", userName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new

```

```

HashMap(), url, reqHeaders, reqParams,
    "PUT");
    String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
    System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void generateRoleAssignmentReport(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentreport";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void generateUserGroupReport(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/usergroupreport";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =

```

```

CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
    System.out.println(jobStatus);
} catch (Exception e) {
    e.printStackTrace();
}
}

public void addGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void removeGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
            "DELETE");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void generateInvalidLoginReport(String fromDate, String
toDate, String fileName) {

```

```

        try {
            String url = this.serverUrl + "/interop/rest/security/" +
                apiVersion + "/invalidloginreport";
            Map<String, String> reqHeaders = new HashMap<String,
                String>();
            reqHeaders.put("Authorization", "Basic " +
                DatatypeConverter
                    .printBase64Binary((this.userName + ":" +
                this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
                String>();
            reqParams.put("from_date", fromDate);
            reqParams.put("to_date", toDate);
            reqParams.put("filename", fileName);

            Map<String, String> restResult =
                CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                    "POST");
            String jobStatus =
                CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void generateRoleAssignmentAuditReport(String fromDate,
        String toDate, String fileName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
                apiVersion + "/roleassignmentauditreport";
            Map<String, String> reqHeaders = new HashMap<String,
                String>();
            reqHeaders.put("Authorization", "Basic " +
                DatatypeConverter
                    .printBase64Binary((this.userName + ":" +
                this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
                String>();
            reqParams.put("from_date", fromDate);
            reqParams.put("to_date", toDate);
            reqParams.put("filename", fileName);

            Map<String, String> restResult =
                CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                    "POST");
            String jobStatus =
                CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

public void updateUsers(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
                this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "UPDATE_USERS");

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
            HashMap(), url, reqHeaders, reqParams,
                "PUT");
        String jobStatus =
            CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static class CSSRESTHelper {
    public static final String REST_CALL_STATUS = "REST_CALL_STATUS";
    public static final String REST_CALL_RESPONSE = "REST_CALL_RESPONSE";

    private static Map<String, String> callRestApi(Map context, String
        url, Map<String, String> requestHeaders,
            Map<String, String> requestParams, String methodType) {
        HttpURLConnection urlConnection = null;
        Map<String, String> restResult = new HashMap<String, String>();
        restResult.put(REST_CALL_STATUS, "-1");
        boolean isPostMethod = "POST".equalsIgnoreCase(methodType) ||
            "PUT".equalsIgnoreCase(methodType);
        try {
            URI baseUri = new URI(url);
            URI uri = null;
            String reqParams = (requestParams != null ?
                buildRequestParams(context, requestParams, isPostMethod)
                    : null);
            if (isPostMethod) {
                uri = new URI(baseUri.getScheme(),
                    baseUri.getAuthority(), baseUri.getPath(), null, null);
            } else {
                uri = new URI(baseUri.getScheme(),
                    baseUri.getAuthority(), baseUri.getPath(), reqParams, null);
            }

            urlConnection = (HttpURLConnection)
                uri.toURL().openConnection();
            urlConnection.setRequestMethod(methodType);

```



```

        if (requestHeaders != null) {
            Set<String> requestHeaderKeys =
requestHeaders.keySet();
            for (String requestHeaderKey : requestHeaderKeys) {

urlConnection.setRequestProperty(requestHeaderKey,
requestHeaders.get(requestHeaderKey));
            }
        }

        urlConnection.setUseCaches(false);
        urlConnection.setDoOutput(true);
        urlConnection.setDoInput(true);

        if (isPostMethod) {
            OutputStreamWriter writer = new
OutputStreamWriter(urlConnection.getOutputStream(),
                Charset.defaultCharset());
            writer.write(reqParams);
            writer.flush();
        }

        if (!isPostMethod) {
            urlConnection.connect();
        }

        int status = urlConnection.getResponseCode();
        restResult.put(REST_CALL_STATUS,
String.valueOf(status));
        String response = readResponse(context,
            (status >= 400 ?
urlConnection.getErrorStream() : urlConnection.getInputStream()));
        restResult.put(REST_CALL_RESPONSE, response);
    } catch (Exception e) {
        restResult.put(REST_CALL_RESPONSE, e.getMessage());
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
    }
    return restResult;
}

private static String buildRequestParams(Map context,
Map<String, String> requestParams, boolean isPostMethod) {
    String reqParams = null;
    try {
        StringBuilder result = new StringBuilder();
        Set<String> reqParamKeys = requestParams.keySet();
        boolean first = true;
        for (String reqParamKey : reqParamKeys) {
            if (first)
                first = false;
            else
                result.append("&");
        }
    }
}

```

```

        String reqParamValue = requestParams.get(reqParamKey);
        result.append((isPostMethod ?
URLLEncoder.encode(reqParamKey, "UTF-8") : reqParamKey));
        result.append("=");
        result.append((isPostMethod ?
URLLEncoder.encode(reqParamValue, "UTF-8") : reqParamValue));
    }
    reqParams = result.toString();
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
return reqParams;
}

private static String readResponse(Map context, InputStream
urlInStream) {
    BufferedReader br = null;
    String response = "";
    try {
        String line;
        br = new BufferedReader(new InputStreamReader(urlInStream,
Charset.defaultCharset()));
        while ((line = br.readLine()) != null) {
            response += line;
        }
    } catch (Exception e) {
        response += e.getMessage();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return response;
}

private static String getCSSRESTJobUrlFromResponse(String response) {
    String jobUrl = "";
    try {
        JSONObject jsonResponse = new JSONObject(response);
        JSONArray links = (JSONArray) jsonResponse.get("links");
        JSONObject jobStatusLink = (JSONObject) links.get(1);
        jobUrl = jobStatusLink.get("href").toString();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return jobUrl;
}

private static String getCSSRESTJobStatusFromResponse(String
response) {
    String jobStatus = "";

```

```

        try {
            JSONObject jsonResponse = new JSONObject(response);
            jobStatus = jsonResponse.get("status").toString();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return jobStatus;
    }

    private static String
    getCSSRESTJobCompletionStatus(Map<String, String> restResult,
    Map<String, String> reqHeader) {
        String completionStatus = "";
        try {
            String restStatus =
restResult.get(CSSRESTHelper.REST_CALL_STATUS);
            if (restStatus.equalsIgnoreCase("200")) {
                String jobUrl =
getCSSRESTJobUrlFromResponse(restResult.get(CSSRESTHelper.REST_CALL_RES
PONSE));

                String restJobStatus = "-1";
                Map<String, String> jobStatusResult = null;
                while (restJobStatus.equalsIgnoreCase("-1")) {
                    jobStatusResult =
CSSRESTHelper.callRestApi(new HashMap(), jobUrl, reqHeader, null,
"GET");

                    String jobStatusStatus =
jobStatusResult.get(CSSRESTHelper.REST_CALL_STATUS);
                    if (jobStatusStatus.equalsIgnoreCase("200")) {
                        restJobStatus =
getCSSRESTJobStatusFromResponse(
jobStatusResult.get(CSSRESTHelper.REST_CALL_RESPONSE));
                    }
                    Thread.sleep(1000);
                }
                completionStatus =
jobStatusResult.get(CSSRESTHelper.REST_CALL_RESPONSE);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return completionStatus;
    }
};
}

```

C

Common Helper Functions for cURL

This appendix shows the common helper functions for cURL for the EPM Cloud REST APIs.

Note: the `USERNAME` variable is `<domain>.<username>`. See [Authentication](#).

```
#!/bin/sh

SERVER_URL="https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/"
USERNAME=<username>
PASSWORD=<password>
APP_NAME="Vision"
API_VERSION="11.1.2.3.600"

funcRemoveTempFiles() {
    for var in "$@"
    do
        if [ -f $var ]; then
            rm $var
        fi
    done
}

funcPrintErrorDetails() {
    contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -d
[:space:]`
    if [ ! -z $contentType ] && [[ $contentType = *"application/json"* ]];
then
        output=`cat $1`
        error=`echo $output | jq '.details'`
        echo "Error details: " $error
    fi
}

funcExecuteRequest() {
    if [ ! -z "$4" ]; then
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $4" -d $3 $2`
    else
        statusCode=`curl -X $1 -s -w "%{http_code}" -u "$USERNAME:$PASSWORD"
-o "response.txt" -D "respHeader.txt" -H "Content-Type: $3" $2`
    fi
    if [ $statusCode != 200 ]; then
        echo "Error executing request"
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails "response.txt"
            funcRemoveTempFiles "respHeader.txt" "response.txt"
        fi
    fi
}
```

```

        exit 0
    fi
}

funcGetStatus() {
    output=`cat response.txt`
    count=`echo $output | jq '.links | length`
    i=0
    pingUrl=""
    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links[$i].rel`
        rel=`echo "$rel" | tr -d "\"`
        if [ "$rel" == "Job Status" ]; then
            pingUrl=`echo $output | jq '.links[$i].href`
            pingUrl=`echo "$pingUrl" | tr -d "\"`
        fi
        i=`expr $i + 1`
    done
    echo $pingUrl
    completed="false"
    while [ $completed != "true" ]; do
        statusCode2=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "pingResponse.txt" -H "Content-Type:
application/x-www-form-urlencoded" "$pingUrl`
        if [ $statusCode2 == 200 ]; then
            status2=`jq '.status' pingResponse.txt`
            if [ $status2 != -1 ]; then
                completed="true"
                echo "Job completed"
            else
                echo "Please wait..."
                sleep 20
            fi
        else
            echo "Please wait..."
            sleep 20
        fi
        funcRemoveTempFiles "pingResponse.txt"
    done
}

funcGetMigrationStatus() {
    output=`cat response.txt`
    count=`echo $output | jq '.links | length`
    i=0
    pingUrl=""
    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links[$i].rel`
        rel=`echo "$rel" | tr -d "\"`
        if [ "$rel" == "Job Status" ]; then
            pingUrl=`echo $output | jq '.links[$i].href`
            pingUrl=`echo "$pingUrl" | tr -d "\"`
        fi
        i=`expr $i + 1`
    done
}

```

```

done
echo $pingUrl
completed="false"
while [ $completed != "true" ]; do
    statusCode2=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "pingResponse.txt" -H "Content-Type: application/x-
www-form-urlencoded" "$pingUrl"`
    if [ $statusCode2 == 200 ]; then
        status2=`jq '.status' pingResponse.txt`
        if [ $status2 == 0 ]; then
            completed="true"
            echo "Job completed"
        elif [ $status2 == 1 ]; then
            output1=`cat pingResponse.txt`
            echo "Error occurred"
            count=`echo $output1 | jq '.items | length'`
            if [ $count == 0 ]; then
                echo `echo $output1 | jq '.details'`
            else
                i=0
                while [ $i -lt $count ]; do
                    echo "Source : " `echo $output1 | jq
'.items['$i'].source'`
                    echo "Destination : " `echo $output1 | jq
'.items['$i'].destination'`
                    firstPing=`echo $output1 | jq
'.items['$i'].links[0].href'`
                    echo ""
                    taskCompleted="false"
                    while [ $taskCompleted != "true" ]; do
                        statusCode3=`curl -X "GET" -s -w "%{http_code}" -
u "$USERNAME:$PASSWORD" -o "taskpingResponse.txt" -H "Content-Type:
application/x-www-form-urlencoded" "$firstPing"`
                        echo $statusCode3
                        output2=`cat taskpingResponse.txt`
                        count1=`echo $output2 | jq '.items | length'`
                        j=0
                        currentMessageCategory=""
                        while [ $j -lt $count1 ]; do
                            msgCategory=`echo $output1 | jq
'.items['$i'].msgCategory'`
                            if [ !-z $currentMessageCategory ] ||
[ $currentMessageCategory != $msgCategory ]; then
                                currentMessageCategory=msgCategory
                                echo $currentMessageCategory
                            fi
                            echo `echo $output2 | jq
'.items['$i'].artifact'` " - " `echo $output2 | jq '.items['$i'].msgText'`
                            count2=`echo $output | jq '.links | length'`
                            k=0
                            firstPing=""
                            while [ $k -lt $count ]; do
                                rel=`echo $output2 | jq
'.links['$i'].rel'`
                                rel=`echo "$rel" | tr -d "\""`

```

```

                                if [ "$rel" == "next" ]; then
                                    firstPing=`echo $output2 |
jq '.links['$i'].href'`
                                    firstPing=`echo
"$firstPing" | tr -d "\""`
                                fi
                                k=`expr $k + 1`
                            done
                            if [ -z $firstPing ]; then
                                taskCompleted="true"
                            fi
                            j=`expr $j + 1`
                        done
                    done
                    i=`expr $i + 1`
                done
            fi
        else
            echo "Please wait..."
            sleep 20
        fi

    else
        echo "Please wait..."
        sleep 20
    fi
    funcRemoveTempFiles "pingResponse.txt" "taskpingResponse.txt"
done
}

funcGetLCMVersions() {
    url=$SERVER_URL/interop/rest
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "List of versions :"
        count=`echo $output | jq '.items | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Version : " `echo $output | jq
'.items['$i'].version'`
            echo "Lifecycle : " `echo $output | jq
'.items['$i'].lifecycle'`
            echo "Latest : " `echo $output | jq '.items['$i'].latest'`
            echo "Link : " `echo $output | jq
'.items['$i'].links[0].href'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
}

```

```

    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGetLCMVersionDetails() {
    url=$SERVER_URL/interop/rest/$API_VERSION
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Version $API_VERSION details :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links['$i'].rel'`
            echo "URL : " `echo $output | jq '.links['$i'].href'`
            echo "Action : " `echo $output | jq '.links['$i'].action'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGetServices() {
    url=$SERVER_URL/interop/rest/$API_VERSION/services
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Services list :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            rel=`echo $output | jq '.links['$i'].rel'`
            rel=`echo "$rel" | tr -d "\""`
            if [ "$rel" != "self" ]; then
                echo "Service : " `echo $output | jq '.links['$i'].rel'`
                echo "URL : " `echo $output | jq '.links['$i'].href'`
                echo "Action : " `echo $output | jq '.links['$i'].action'`
                echo ""
            fi
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```



```

funcRecreateService() {
    echo "Are you sure you want to recreate the EPM environment (yes/
no): no ?[Press Enter]"
    read toCreate
    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$API_VERSION/services/$1/recreate
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started recreating the environment successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGetApplicationSnapshotDetails() {
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Application details :"
        echo "Application snapshot name : " `echo $output | jq
'.items[0].name'`
        echo "Application snapshot type : " `echo $output | jq
'.items[0].type'`
        echo "Can be exported flag : " `echo $output | jq
'.items[0].canExport'`
        echo "Can be imported flag : " `echo $output | jq
'.items[0].canImport'`
        echo "Can be uploaded flag : " `echo $output | jq
'.items[0].canUpload'`
        echo "Can be downloaded flag : " `echo $output | jq
'.items[0].canDownload'`
        count=`echo $output | jq '.links | length'`
        i=0
        echo "Services details :"
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links['$i'].rel'`
            echo "URL : " `echo $output | jq '.links['$i'].href'`
            echo "Action : " `echo $output | jq '.links['$i'].action'`
            echo ""
        done
    fi
}

```

```

        i=`expr $i + 1`
    done
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcListFiles() {
    url=$SERVER_URL/interop/rest/$API_VERSION/applicationsnapshots
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    list=`cat response.txt | jq 'select(.items != null) | .items[].name'`
    if [[ ! -z $list ]]; then
        echo $list
    else
        echo "No files found"
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcDeleteFile() {
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName
    funcExecuteRequest "DELETE" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Deleted successfully"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcUploadFile() {
    infile=$1
    if [ ! -f $infile ]; then
        echo "File does not exist"
        exit 0
    fi
    encodedFileName=$(echo $infile | sed -f urlencode.sed)
    url="$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/contents?q="
    filename=$( basename $infile)
    filesize=$( stat -c %s $infile)
    bs=52428800
    noOfPackets=$(( (filesize / bs) + 1) )
    uploadedsize=0
    isFirst=true
}

```

```

count=1
isLast="false"
if [ $noOfPackets = 1 ]; then
    isLast="true"
fi
tempFile=/u01/temp/$filename

if [ ! -d "/u01/temp" ]; then
    mkdir /u01/temp
fi

while [ $uploadedsize -ne $filesize ]
do
    skip=$uploadedsize
    temp=$((filesize - uploadedsize))
    if [ $temp -le $bs ]; then
        length=$temp
    else
        length=$bs
    fi
    echo "Skip : $skip"
    echo "Length : $length"

    (
        dd bs=1 skip=$skip count=0 &> /dev/null
        dd bs=$length count=1 of=$tempFile &> /dev/null
    ) < "$infile"

    param=$(echo
"{chunkSize=$length,isFirst=$isFirst,isLast=$isLast}" | sed -f
urlencode.sed)
    urlwithparam="$url$params"
    echo $urlwithparam
    statusCode=`curl -X POST -s -w "%{http_code}" -T $tempFile -u
"$USERNAME:$PASSWORD" -o "response.txt" -D "respHeader.txt" -H
"Content-Type: application/octet-stream" "$urlwithparam"`

    funcRemoveTempFiles $tempFile

    if [ $statusCode == 200 ]; then
        output=`cat response.txt`
        status=`echo $output | jq '.status'`
        if [ $status -gt 0 ]; then
            error=`echo $output | jq '.details'`
            echo "Error occurred. " $error
            funcRemoveTempFiles "respHeader.txt" "response.txt"
            exit 0
        else if [ $status == -1 ] || [ $isLast == "true" ]; then
            funcGetStatus "GET"
        fi
    fi
else
    echo "Error executing request"
    if [ $statusCode != 000 ]; then
        echo "Response error code : " $statusCode
    fi
fi

```

```

        funcPrintErrorDetails "response.txt"
        funcRemoveTempFiles "respHeader.txt" "response.txt"
    fi
    exit 0
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
uploadedsize=$((uploadedsize + length))
isFirst="false"
echo "isFirst : $isFirst"
count=$((count + 1))
if [ $count = $noOfPackets ]; then
    isLast="true"
fi
echo "Uploaded Size : $uploadedsize"
echo "isLast : $isLast"
done

echo "Uploaded File Successfully"
}

funcDownloadFile() {
    filepath="/u01/$1"
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/contents
    statusCode=`curl -X GET -s -w "%{http_code}" -u "$USERNAME:$PASSWORD" -
o $filepath -H "Content-Type: application/x-www-form-urlencoded" -D
respHeader.txt $url`

    if [ $statusCode == 200 ]; then
        contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -d
[:space:]`
        if [ ! -z $contentType ] && [[ $contentType = *"application/
json"* ]]; then
            output=`cat $filepath`
            error=`echo $output | jq '.details'`
            echo "Error occurred. " $error
            funcRemoveTempFiles $filepath
        else
            fileExtension=`echo $(grep -r "fileExtension: " respHeader.txt |
awk '{print ($2)}') | tr -d [:space:]`
            if [ ! -z $fileExtension ]; then
                if [[ ! $filepath =~ \.${fileExtension}$ ]]; then
                    mv $filepath $filepath.${fileExtension}
                fi
            fi
            echo "Downloaded file successfully"
        fi
    else
        echo "Error listing files. "
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails $filepath
            funcRemoveTempFiles $filepath
        fi
    fi
}

```

```

    fi
    funcRemoveTempFiles "respHeader.txt"
}

funcImportSnapshot() {
    param=$(echo "{type:import}" | sed -f urlencode.sed)
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/migration?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started importing successfully"
        funcGetMigrationStatus "POST"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcExportSnapshot() {
    param=$(echo "{type:export}" | sed -f urlencode.sed)
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/migration?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started exporting succesfully"
        funcGetMigrationStatus "POST"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcProvideFeedback() {
    url=$SERVER_URL/interop/rest/$LCM_VERSION/feedback
    description=$(echo $1 | sed -f urlencode.sed)
    param="{\"configuration\":
{\"URL\": \"$SERVER_URL\"}, \"description\": \"$description\"}"
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Feedback successful"
    else
        error=`echo $output | jq '.details'`

```

```

        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcHardReset() {
    echo "Are you sure you want to restart the service instance (yes/no):
no ?[Press Enter] "
    read toCreate
    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi

    url=$SERVER_URL/interop/rest/$LCM_VERSION/services/PBCS/resetservice
    comment=$(echo $1 | sed -f urlencode.sed)
    param="{\"comment\": \"${comment}\"}"
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started hard reset succesfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGenerateAuditReport(){
    param=$(echo "{type:userauditreport,fileName:$1,since:$2,until:$3}" |
sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/reports?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started generating report successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGenerateProvisionReport(){
    param=$(echo "{type:provisionreport,fileName:$1}" | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/reports?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`

```

```

status=`echo $output | jq '.status'`
if [ $status == -1 ]; then
    echo "Started generating report successfully"
    funcGetStatus "GET"
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcExecuteJob() {
    url="$SERVER_URL/HyperionPlanning/rest/$API_VERSION/
applications/$APP_NAME/jobs"
    encodedJobName=$(echo $2 | sed -f urlencode.sed)
    if [ ! -z "$3" ]; then
        param="jobType=$1&jobName=$encodedJobName&parameters=$3"
    else
        param="jobType=$1&jobName=$encodedJobName"
    fi
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started executing job successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcIntegrationScenarioImportMetadataIntoApplication() {
    funcUploadFile "DemoApplication_HSS_Vision.zip"
    funcExecuteJob "IMPORT_METADATA" "accountMetadata"
    "{importZipFileName=accounts.zip}"
    funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}

funcIntegrationScenarioImportDataRunCalcCopyToAso() {
    funcUploadFile "data.csv"
    funcExecuteJob "IMPORT_DATA" "loadingqldata"
    "{importFileName=data.csv}"
    funcExecuteJob "CUBE_REFRESH", "cubeRefresh"
    funcExecuteJob "PLAN_TYPE_MAP" "CampaignToReporting"
    "{clearData=false}"
}

funcIntegrationScenarioExportMetadataAndDataAndDownloadFiles() {
    funcExecuteJob "EXPORT_METADATA" "exportentitymetadata"
    "{exportZipFileName=entitydata.zip}"
    funcExecuteJob "EXPORT_DATA" "Forecastdata"
}

```

```

"{exportFileName=forecastdata.zip}"
  funcListFiles
  funcDownloadFile "entitydata.zip"
  funcDownloadFile "forecastdata.zip"
}

funcIntegrationScenarioRemoveUnnecessaryFiles() {
  funcListFiles
  funcDeleteFile "entitymetadata.csv"
  funcDeleteFile "forecastdata.csv"
}

funcIntegrationScenarioExportDataAndDownloadFiles() {
  funcExecuteJob "EXPORT_DATA" "entitydata"
  "{exportFileName:entitydata.zip}"
  funcExecuteJob "EXPORT_DATA" "forecastdata"
  "{exportFileName:forecastdata.zip}"
  funcListFiles
  funcDownloadFile "entitydata.zip"
  funcDownloadFile "forecastdata.zip"
}

funcIntegrationScenarioRefreshTheApplication() {
  funcUploadFile "accounts.zip"
  funcExecuteJob "IMPORT_METADATA" "accountMetadata"
  "{importZipFileName:accounts.zip}"
  funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}

funcIntegrationScenarioCloneServiceInstance() {
  # Part 1 : Change SERVER_URL, USERNAME, PASSWORD, API_VERSION variables
  values to match those of first environment
  # Download file from source instance.
  # Comment out all lines below Part 2
  # Uncomment the below line for the first step.
  # funcDownloadFile "Artifact Snapshot"

  # Part 2 : Change SERVER_URL, USERNAME, PASSWORD, API_VERSION to match
  those of second environment.
  # Clone the service instance.
  # Comment out code for download file.
  # Uncomment below lines
  funcRecreateService "PBCS"
  funcDeleteFile "Artifact Snapshot"
  funcUploadFile "Artifact Snapshot.zip"
  funcImportSnapshot "Artifact Snapshot"
}

funcIntegrationScenarioImportMetadataIntoApplication
funcIntegrationScenarioImportDataRunCalcCopyToAso
funcIntegrationScenarioExportMetadataAndDataAndDownloadFiles
funcIntegrationScenarioRemoveUnnecessaryFiles
funcIntegrationScenarioExportDataAndDownloadFiles
funcIntegrationScenarioRefreshTheApplication#!/bin/sh

```



```

SERVER_URL="https://<SERVICE_NAME>-
<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/"
USERNAME="epm_default_cloud_admin"
PASSWORD="epm_cloud"
APP_NAME="Vision"
API_VERSION="11.1.2.3.600"

funcRemoveTempFiles() {
    for var in "$@"
    do
        if [ -f $var ]; then
            rm $var
        fi
    done
}

funcPrintErrorDetails() {
    contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -d
[:space:]`
    if [ ! -z $contentType ] && [[ $contentType = *"application/
json"* ]]; then
        output=`cat $1`
        error=`echo $output | jq '.details'`
        echo "Error details: " $error
    fi
}

funcExecuteRequest() {
    if [ ! -z "$4" ]; then
        statusCode=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "response.txt" -D "respHeader.txt" -H
"Content-Type: $4" -d $3 $2`
    else
        statusCode=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "response.txt" -D "respHeader.txt" -H
"Content-Type: $3" $2`
    fi
    if [ $statusCode != 200 ]; then
        echo "Error executing request"
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails "response.txt"
            funcRemoveTempFiles "respHeader.txt" "response.txt"
        fi
    fi
    exit 0
}

funcGetStatus() {
    output=`cat response.txt`
    count=`echo $output | jq '.links | length'`
    i=0
    pingUrl=""
    while [ $i -lt $count ]; do
        rel=`echo $output | jq '.links[$i].rel'`

```

```

rel=`echo "$rel" | tr -d "\"`
if [ "$rel" == "Job Status" ]; then
    pingUrl=`echo $output | jq '.links['$i'].href'`
    pingUrl=`echo "$pingUrl" | tr -d "\"`
fi
i=`expr $i + 1`
done
echo $pingUrl
completed="false"
while [ $completed != "true" ]; do
    statusCode2=`curl -X $1 -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o "pingResponse.txt" -H "Content-Type: application/x-
www-form-urlencoded" "$pingUrl"`
    if [ $statusCode2 == 200 ]; then
        status2=`jq '.status' pingResponse.txt`
        if [ $status2 != -1 ]; then
            completed="true"
            echo "Job completed"
        else
            echo "Please wait..."
            sleep 20
        fi
    else
        echo "Please wait..."
        sleep 20
    fi
    funcRemoveTempFiles "pingResponse.txt"
done
}

funcGetLCMVersions() {
url=$SERVER_URL/interop/rest
funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

output=`cat response.txt`
status=`echo $output | jq '.status'`
if [ $status == 0 ]; then
    echo "List of versions :"
    count=`echo $output | jq '.items | length'`
    i=0
    while [ $i -lt $count ]; do
        echo "Version : " `echo $output | jq '.items['$i'].version'`
        echo "Lifecycle : " `echo $output | jq '.items['$i'].lifecycle'`
        echo "Latest : " `echo $output | jq '.items['$i'].latest'`
        echo "Link : " `echo $output | jq '.items['$i'].links[0].href'`
        echo ""
        i=`expr $i + 1`
    done
else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

```

funcGetLCMVersionDetails() {
    url=${SERVER_URL}/interop/rest/$API_VERSION
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Version $API_VERSION details :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links[$i].rel'`
            echo "URL : " `echo $output | jq '.links[$i].href'`
            echo "Action : " `echo $output | jq '.links[$i].action'`
            echo ""
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGetServices() {
    url=${SERVER_URL}/interop/rest/$API_VERSION/services
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Services list :"
        count=`echo $output | jq '.links | length'`
        i=0
        while [ $i -lt $count ]; do
            rel=`echo $output | jq '.links[$i].rel'`
            rel=`echo "$rel" | tr -d "\""`
            if [ "$rel" != "self" ]; then
                echo "Service : " `echo $output | jq
'.links[$i].rel'`
                echo "URL : " `echo $output | jq '.links[$i].href'`
                echo "Action : " `echo $output | jq
'.links[$i].action'`
                echo ""
            fi
            i=`expr $i + 1`
        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

```

```

funcRecreateService() {
    echo "Are you sure you want to recreate the EPM environment (yes/no):
no ?[Press Enter]"
    read toCreate
    if [ $toCreate != "yes" ]; then
        echo "User cancelled the recreate command"
        exit 0
    fi

    url=${SERVER_URL}/interop/rest/$API_VERSION/services/$1/recreate
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started recreating the environment successfully"
        funcGetStatus "GET"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcGetApplicationSnapshotDetails() {
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=${SERVER_URL}/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Application details :"
        echo "Application snapshot name : " `echo $output | jq
'.items[0].name'`
        echo "Application snapshot type : " `echo $output | jq
'.items[0].type'`
        echo "Can be exported flag : " `echo $output | jq
'.items[0].canExport'`
        echo "Can be imported flag : " `echo $output | jq
'.items[0].canImport'`
        echo "Can be uploaded flag : " `echo $output | jq
'.items[0].canUpload'`
        echo "Can be downloaded flag : " `echo $output | jq
'.items[0].canDownload'`
        count=`echo $output | jq '.links | length'`
        i=0
        echo "Services details :"
        while [ $i -lt $count ]; do
            echo "Service : " `echo $output | jq '.links[$i].rel'`
            echo "URL : " `echo $output | jq '.links[$i].href'`
            echo "Action : " `echo $output | jq '.links[$i].action'`
            echo ""
            i=`expr $i + 1`
        done
    fi
}

```

```

        done
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcListFiles() {
    url=$SERVER_URL/interop/rest/$API_VERSION/applicationsnapshots
    funcExecuteRequest "GET" $url "application/x-www-form-urlencoded"

    list=`cat response.txt | jq 'select(.items != null)
| .items[].name'`
    if [[ ! -z $list ]]; then
        echo $list
    else
        echo "No files found"
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcDeleteFile() {
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName
    funcExecuteRequest "DELETE" $url "application/x-www-form-
urledcoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == 0 ]; then
        echo "Deleted successfully"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcUploadFile() {
    infile=$1
    if [ ! -f $infile ]; then
        echo "File does not exist"
        exit 0
    fi
    encodedFileName=$(echo $infile | sed -f urlencode.sed)
    url="$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/contents?q="
    filename=$( basename $infile)
    filesize=$( stat -c %s $infile)
    bs=52428800
    noOfPackets=$(( (filesize / bs) + 1) )
    uploadedsize=0

```

```

isFirst=true
count=1
isLast="false"
if [ $noOfPackets = 1 ]; then
    isLast="true"
fi
tempFile=/u01/temp/$filename

if [ ! -d "/u01/temp" ]; then
    mkdir /u01/temp
fi

while [ $uploadedsize -ne $filesize ]
do
    skip=$uploadedsize
    temp=$((filesize - uploadedsize))
    if [ $temp -le $bs ]; then
        length=$temp
    else
        length=$bs
    fi
    echo "Skip : $skip"
    echo "Length : $length"

    (
        dd bs=1 skip=$skip count=0 &> /dev/null
        dd bs=$length count=1 of=$tempFile &> /dev/null
    ) < "$infile"

    param=$(echo "{chunkSize=$length,isFirst=$isFirst,isLast=$isLast}" |
sed -f urlencode.sed)
    urlwithparam="$url$param"
    echo $urlwithparam
    statusCode=`curl -X POST -s -w "%{http_code}" -T $tempFile -u
"$USERNAME:$PASSWORD" -o "response.txt" -D "respHeader.txt" -H "Content-
Type: application/octet-stream" "$urlwithparam"`

    funcRemoveTempFiles $tempFile

    if [ $statusCode == 200 ]; then
        output=`cat response.txt`
        status=`echo $output | jq '.status'`
        if [ $status != 0 ]; then
            error=`echo $output | jq '.details'`
            echo "Error occurred. " $error
            funcRemoveTempFiles "respHeader.txt" "response.txt"
            exit 0
        fi
    else
        echo "Error executing request"
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails "response.txt"
            funcRemoveTempFiles "respHeader.txt" "response.txt"
        fi
    fi

```

```

        exit 0
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
    uploadedsize=$((uploadedsize + length))
    isFirst="false"
    echo "isFirst : $isFirst"
    count=$((count + 1))
    if [ $count = $noOfPackets ]; then
        isLast="true"
    fi
    echo "Uploaded Size : $uploadedsize"
    echo "isLast : $isLast"
done

echo "Uploaded File Successfully"
}

funcDownloadFile() {
    filepath="/u01/$1"
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
    applicationsnapshots/$encodedFileName/contents
    statusCode=`curl -X GET -s -w "%{http_code}" -u
"$USERNAME:$PASSWORD" -o $filepath -H "Content-Type: application/x-www-
form-urlencoded" -D respHeader.txt $url`

    if [ $statusCode == 200 ]; then
        contentType=`echo $(grep 'Content-Type:' respHeader.txt) | tr -
d [:space:]`
        if [ ! -z $contentType ] && [[ $contentType = *"application/
json"* ]]; then
            output=`cat $filepath`
            error=`echo $output | jq '.details'`
            echo "Error occurred. " $error
            funcRemoveTempFiles $filepath
        else
            fileExtension=`echo $(grep -r "fileExtension: "
respHeader.txt | awk '{print ($2)}') | tr -d [:space:]`
            if [ ! -z $fileExtension ]; then
                if [[ ! $filepath =~ \.${fileExtension}$ ]]; then
                    mv $filepath $filepath.${fileExtension}
                fi
            fi
            echo "Downloaded file successfully"
        fi
    else
        echo "Error listing files. "
        if [ $statusCode != 000 ]; then
            echo "Response error code : " $statusCode
            funcPrintErrorDetails $filepath
            funcRemoveTempFiles $filepath
        fi
    fi
    funcRemoveTempFiles "respHeader.txt"
}

```

```

funcImportSnapshot() {
    param=$(echo "{type:import}" | sed -f urlencode.sed)
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/migration?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started importing successfully"
        funcGetStatus "POST"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcExportSnapshot() {
    param=$(echo "{type:export}" | sed -f urlencode.sed)
    encodedFileName=$(echo $1 | sed -f urlencode.sed)
    url=$SERVER_URL/interop/rest/$API_VERSION/
applicationsnapshots/$encodedFileName/migration?q=$param
    funcExecuteRequest "POST" $url "application/x-www-form-urlencoded"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started exporting successfully"
        funcGetStatus "POST"
    else
        error=`echo $output | jq '.details'`
        echo "Error occurred. " $error
    fi
    funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcExecuteJob() {
    url="$SERVER_URL/HyperionPlanning/rest/$API_VERSION/
applications/$APP_NAME/jobs"
    encodedJobName=$(echo $2 | sed -f urlencode.sed)
    if [ ! -z "$3" ]; then
        param="jobType=$1&jobName=$encodedJobName&parameters=$3"
    else
        param="jobType=$1&jobName=$encodedJobName"
    fi
    funcExecuteRequest "POST" $url $param "application/json"

    output=`cat response.txt`
    status=`echo $output | jq '.status'`
    if [ $status == -1 ]; then
        echo "Started executing job successfully"
        funcGetStatus "GET"
    fi
}

```



```

else
    error=`echo $output | jq '.details'`
    echo "Error occurred. " $error
fi
funcRemoveTempFiles "respHeader.txt" "response.txt"
}

funcIntegrationScenarioImportMetadataIntoApplication() {
    funcUploadFile "DemoApplication_HSS_Vision.zip"
    funcExecuteJob "IMPORT_METADATA" "accountMetadata"
    "{importZipFileName=accounts.zip}"
    funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}

funcIntegrationScenarioImportDataRunCalcCopyToAso() {
    funcUploadFile "data.csv"
    funcExecuteJob "IMPORT_DATA" "loadingqldata"
    "{importFileName=data.csv}"
    funcExecuteJob "CUBE_REFRESH", "cubeRefresh"
    funcExecuteJob "PLAN_TYPE_MAP" "CampaignToReporting"
    "{clearData=false}"
}

funcIntegrationScenarioExportMetadataAndDataAndDownloadFiles() {
    funcExecuteJob "EXPORT_METADATA" "exportentitymetadata"
    "{exportZipFileName=entitydata.zip}"
    funcExecuteJob "EXPORT_DATA" "Forecastdata"
    "{exportFileName=forecastdata.zip}"
    funcListFiles
    funcDownloadFile "entitydata.zip"
    funcDownloadFile "forecastdata.zip"
}

funcIntegrationScenarioRemoveUnnecessaryFiles() {
    funcListFiles
    funcDeleteFile "entitymetadata.csv"
    funcDeleteFile "forecastdata.csv"
}

funcIntegrationScenarioExportDataAndDownloadFiles() {
    funcExecuteJob "EXPORT_DATA" "entitydata"
    "{exportFileName:entitydata.zip}"
    funcExecuteJob "EXPORT_DATA" "forecastdata"
    "{exportFileName:forecastdata.zip}"
    funcListFiles
    funcDownloadFile "entitydata.zip"
    funcDownloadFile "forecastdata.zip"
}

funcIntegrationScenarioRefreshTheApplication() {
    funcUploadFile "accounts.zip"
    funcExecuteJob "IMPORT_METADATA" "accountMetadata"
    "{importZipFileName:accounts.zip}"
    funcExecuteJob "CUBE_REFRESH" "cubeRefresh"
}

```

```
funcIntegrationScenarioCloneServiceInstance() {
    # Part 1 : Change SERVER_URL, USERNAME, PASSWORD, API_VERSION variables
    values to match those of first environment
    # Download file from source instance.
    # Comment out all lines below Part 2
    # Uncomment the below line for the first step.
    # funcDownloadFile "Artifact Snapshot"

    # Part 2 : Change SERVER_URL, USERNAME, PASSWORD, API_VERSION to match
    those of second environment.
    # Clone the service instance.
    # Comment out code for download file.
    # Uncomment below lines
    funcRecreateService "PBCS"
    funcDeleteFile "Artifact Snapshot"
    funcUploadFile "Artifact Snapshot.zip"
    funcImportSnapshot "Artifact Snapshot"
}

funcIntegrationScenarioImportMetadataIntoApplication
funcIntegrationScenarioImportDataRunCalcCopyToAso
funcIntegrationScenarioExportMetadataAndDataAndDownloadFiles
funcIntegrationScenarioRemoveUnnecessaryFiles
funcIntegrationScenarioExportDataAndDownloadFiles
funcIntegrationScenarioRefreshTheApplication
```

Note on Proxy Setting: In case of proxies, set the proxy host and port as the system arguments.

D

CSS Common Helper Functions for cURL

This appendix shows the CSS common helper functions for cURL for the EPM Cloud REST APIs.

Note on Proxy Setting: In case of proxies, set the proxy host and port as the system arguments.

```
#!/bin/sh
#set -x
export PATH=$PATH:<PATH_TO_JQ_BINARY>
SERVER_URL="<SERVICE_URL>"
USERNAME="<USERNAME>"
PASSWORD="<PASSWORD>"
API_VERSION="v1"

# To avoid SSL connection issue in the environment please add -k option for
below curl commands.
funcCallRESTAPI() {
    if [ "$1" == "GET" ] || [ "$1" == "DELETE" ]; then
        if [ "$6" != "" ]; then
            echo `curl -s -u $4:$5 -H "$3" --request $1 -G $2 -d "$6"`
        else
            echo `curl -s -u $4:$5 -H "$3" --request $1 -G $2`
        fi
    else
        if [ "$6" != "" ]; then
            echo `curl -s -u $4:$5 -H "$3" --request $1 $2 -d
"$6"`
        else
            echo `curl -s -u $4:$5 -H "$3" --request $1 $2`
        fi
    fi
}

funcCSSRESTHelper() {
    jobOutput=$(funcCallRESTAPI "$1" "$2" "$3" "$4" "$5" "$6")
    jobUrl=`echo $jobOutput | jq '.links[1].href'`
    if [ $jobUrl != null ]; then
        jobUrl="${jobUrl%\"}"
        jobUrl="${jobUrl#\"}"
        jobStatus=-1
        while [ $jobStatus == -1 ]; do
            jobOutput=$(funcCallRESTAPI "GET" "$jobUrl"
"$header" "$USERNAME" "$PASSWORD")
            jobStatus=`echo $jobOutput | jq '.status'`
        done
        restStatus=`echo $jobOutput | jq '.details'`
        restStatus="${restStatus%\"}"
        restStatus="${restStatus#\"}"
    fi
}
```

```

        statusMessage=""
        if [ $jobStatus == 0 ]; then
            statusMessage="$7 completed successfully."
#"$restStatus"
        else
            statusMessage=$restStatus
        fi
        echo "$statusMessage"
    else
        failedMessage=`echo $jobOutput | jq '.details'`
        failedMessage="{failedMessage%\"}"
        failedMessage="{failedMessage#\\"}"
        echo $failedMessage
    fi
}

funcAddUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&userpassword=$2&resetpassword=$3"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AddUsers"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="RemoveUsers"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAssignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=ASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AssignRole"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcUnassignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=UNASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="UnassignRole"

```

```

        statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
        echo $statusMessage
    }

funcAddUsersToGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USERS_TO_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
    cssRESTAPI="AddUsersToGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveUsersFromGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=REMOVE_USERS_FROM_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="RemoveUsersFromGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAddUserToGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USER_TO_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
    cssRESTAPI="AddUserToGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveUserFromGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=REMOVE_USER_FROM_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="RemoveUserFromGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateRoleAssignmentReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"

```

```

"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateUserGroupReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
usergroupreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateUserGroupReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAddGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="addGroups"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="removeGroups"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateInvalidLoginReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
invalidloginreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateInvalidLoginReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateRoleAssignmentAuditReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentauditreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-

```

```
urlencoded; charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentAuditReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcUpdateUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=UPDATE_USERS"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="UpdateUsers"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

#funcAddUsers test1.csv password false
#funcRemoveUsers test2.csv
#funcAssignRole test3.csv "Power User"
#funcUnAssignRole test4.csv Viewer
#funcAddUsersToGroup test5.csv TestNativeGroup1
#funcRemoveUsersFromGroup test6.csv TestNativeGroup2
#funcGenerateRoleAssignmentReport RoleAssignmentReport.csv
#funcGenerateUserGroupReport UserGroupReport.csv
#funcAddUserToGroups groups.csv joe
#funcRemoveUserFromGroups groups.csv joe
#funcAddGroups CreateGroup1.csv
#funcRemoveGroups DeleteGroup1.csv
#funcGenerateInvalidLoginReport 2021-06-01 2021-06-10 invalidLoginReport.csv
#funcGenerateRoleAssignmentAuditReport 2021-06-01 2021-06-10
roleAssignmentAuditReport.csv
#funcUpdateUsers updateuser.csv
```

E

CSS Common Helper Functions for Groovy

This appendix shows the CSS common helper functions for Groovy for the EPM Cloud REST APIs.

Note:

- **Proxy setting:** In case of proxies, set the proxy host and port as the system arguments.
- **Username:** The `username` variable uses the format `<domain>.<username>`.
[Authentication](#).

```
import java.nio.charset.StandardCharsets

import groovy.json.JsonSlurper

serverUrl="<SERVICE_URL>"
username="<DOMAINNAME.USERNAME>"
password="<PASSWORD>"

apiVersion = "v1";
userCredentials = username + ":" + password;
basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)

def getResponse(is) {
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line+"\n");
    }
    br.close();
    return sb.toString();
}

def getUrlFromResponse(scenario, response, relValue) {
    def object = new JsonSlurper().parseText(response)
    def pingUrlStr
    if (object.status == -1) {
        println "Started - " + scenario
        def links = object.links
        links.each{
            if (it.rel.equals(relValue)) {
                pingUrlStr=it.href
            }
        }
    } else {
        println "Error details: " + object.details
        System.exit(0);
    }
}
```



```

    }
    return pingUrlStr
}

def getJobStatus(pingUrlString, methodType) {

    def pingUrl = new URL(pingUrlString);
    def completed = false;
    while (!completed) {
        pingResponse = executeRequest(pingUrl, methodType, null,
"application/x-www-form-urlencoded");
        status = getJobStatusFromResponse(pingResponse);
        if (status == "Processing") {
            try {
                println "Processing. Please wait..."
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                completed = true
            }
        } else {
            println status
            completed = true
        }
    }
}

def getJobStatusFromResponse(response) {
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == -1)
        return "Processing"
    else if (status == 0)
        return "Completed"
    else
        return object.details
}

def getJobDetailsFromResponse(response) {
    def object = new JsonSlurper().parseText(response)
    def details = object.details
    if (details != null)
        return object.details
    else
        return null
}

def executeRequest(url, requestType, payload, contentType) {
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
    connection.setDoOutput(true);
    connection.setInstanceFollowRedirects(false);
    connection.setRequestMethod(requestType);
    connection.setRequestProperty("Content-Type", contentType);
    // connection.setRequestProperty("charset",
StandardCharsets.UTF_8);
}

```

```

connection.setRequestProperty("Authorization", basicAuth);
connection.setUseCaches(false);

if (payload != null) {
    OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
    writer.write(payload);
    writer.flush();
}

int statusCode
try {
    statusCode = connection.responseCode;
} catch (all) {
    println "Error connecting to the URL"
    System.exit(0);
}

def response
if (statusCode == 200 || statusCode == 201) {
    if (connection.getContentType() != null && !
connection.getContentType().startsWith("application/json")) {
        println "Error occurred in server"
        System.exit(0)
    }
    InputStream is = connection.getInputStream();
    if (is != null)
        response = getResponse(is)
} else {
    println "Error occurred while executing request"
    println "Response error code : " + statusCode
    InputStream is = connection.getErrorStream();
    if (is != null && connection.getContentType() != null &&
connection.getContentType().startsWith("application/json"))
        println getJobStatusFromResponse(getResponse(is))
    System.exit(0);
}
connection.disconnect();
return response;
}

def addUsersToGroup(fileName, groupName) {

    String scenario = "Adding users in " + fileName + " to group " +
groupName;
    String params = "jobtype=ADD_USERS_TO_GROUP&filename="+ fileName
+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
}

```

```

    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def removeUsersFromGroup(fileName, groupName) {

    String scenario = "Removing users in " + fileName + " from group "
+ groupName;
    String params = "jobtype=REMOVE_USERS_FROM_GROUP&filename="+
fileName+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def addUserToGroups(fileName, userName) {

    String scenario = "Adding users in " + fileName + " to group " +
userName;
    String params = "jobtype=ADD_USER_TO_GROUPS&filename="+ fileName
+"&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

```

```

def removeUserFromGroups(fileName, userName) {

    String scenario = "Removing users in " + fileName + " from group " +
userName;
    String params = "jobtype=REMOVE_USER_FROM_GROUPS&filename="+ fileName
+"&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def addUsers(fileName, resetPassword, userPassword) {

    String scenario = "Creating users in " + fileName;
    String params = "jobtype=ADD_USERS&filename="+ fileName
+"&resetpassword="+ resetPassword + "&userpassword="+ userPassword;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def addUsers(fileName) {
    addUsers(fileName, null, null);
}

def deleteUsers(fileName) {

    String scenario = "Deleting users in " + fileName;
    String params = null;
    def url = null;
    def response = null;

```

```

    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/users?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def assignUsersRoles(fileName, roleName) {

    String scenario = "Assigning users in " + fileName + " with role "
+ roleName;
    String params = "jobtype=ASSIGN_ROLE&filename="+ fileName
+"&rolename="+ roleName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def unassignUsersRoles(fileName, roleName) {

    String scenario = "Un-assigning users in " + fileName + " with
role " + roleName;
    String params = "jobtype=UNASSIGN_ROLE&filename="+ fileName
+"&rolename="+ roleName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
}

```

```

        if (response != null) {
            getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
        }
    }

def generateRoleAssignmentReport(fileName) {

    String scenario = "Generating Role assignment report in " + fileName;
    String params = "jobtype=GENERATE_ROLE_ASSIGNMENT_REPORT&filename="+
fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
roleassignmentreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def generateUserGroupReport(fileName) {

    String scenario = "Generating User Group Report in " + fileName;
    String params = "jobtype=GENERATE_USER_GROUP_REPORT&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
usergroupreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def addGroups(fileName) {
    println "addgroups"
    String scenario = "Creating Groups in " + fileName;
    String params = "filename="+ fileName;
    def url = null;
    def response = null;

```

```

    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def removeGroups(fileName) {

    String scenario = "Deleting Groups in " + fileName;
    String params = null;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/groups?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def generateRoleAssignmentAuditReport(from_date,to_date,fileName) {

    String scenario = "Generating Role assignment audit report in " +
        fileName;
    String params =
"jobtype=GENERATE_ROLE_ASSIGNMENT_AUDIT_REPORT&from_date="+from_date+"&
to_date="+to_date+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/roleassignmentauditreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
    if (response != null) {

```

```

        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def generateInvalidLoginReport(from_date,to_date,fileName) {

    String scenario = "Generating Invalid Login report in " + fileName;
    String params =
"jobtype=GENERATE_INVALID_LOGIN_REPORT&from_date="+from_date+"&to_date="+to_d
ate+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
invalidloginreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def updateUsers(fileName) {

    String scenario = "Updating users from " + fileName ;
    String params = "jobtype=UPDATE_USERS&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

//Execute commands here
//addUsersToGroup("Users.csv",
"G1");
//PUT
//removeUsersFromGroup("Users.csv",
"G1");

```



```

        //PUT
//addUsers("AddUsers123.csv", "false",
"newPassword"); //POST
//
addUsers("AddUsers456.csv");

        //POST
//
deleteUsers("RemoveUsers.csv"); //

DELETE
//assignUsersRoles("Users.csv", "Service
Administrator"); //PUT
//assignUsersRoles("users.csv",
"viewer"); //P
UT
//unassignUsersRoles("Users.csv", "Drill
Through"); //PUT
//
generateRoleAssignmentReport("GroovySampleReport3.csv");
        // POST
//
generateUserGroupReport("UserGroupReportGroovy.csv");
        // POST
//addUserToGroups("Group.csv",
"user1"); //PUT
//removeUserFromGroups("groups.csv",
"joe"); //PUT
//
addGroups("CreateGroup1.csv");
        // POST
//
removeGroups("DeleteGroup1.csv");
        // DELETE
//generateInvalidLoginReport("2020-06-01", "2021-06-10",
"report12345.csv"); //POST
//updateUsers("updateuser.csv"); // PUT

```

F

REST API Examples with Postman

This appendix provides examples of how to run selected REST APIs using a web client called Postman.

- [Example: Using REST APIs to Upload with Postman](#)
- [Example: Using REST APIs to Upload to an External Directory with Postman](#)
- [Example: Using REST APIs to Upload a Snapshot with Postman](#)

Example: Using REST APIs to Upload with Postman

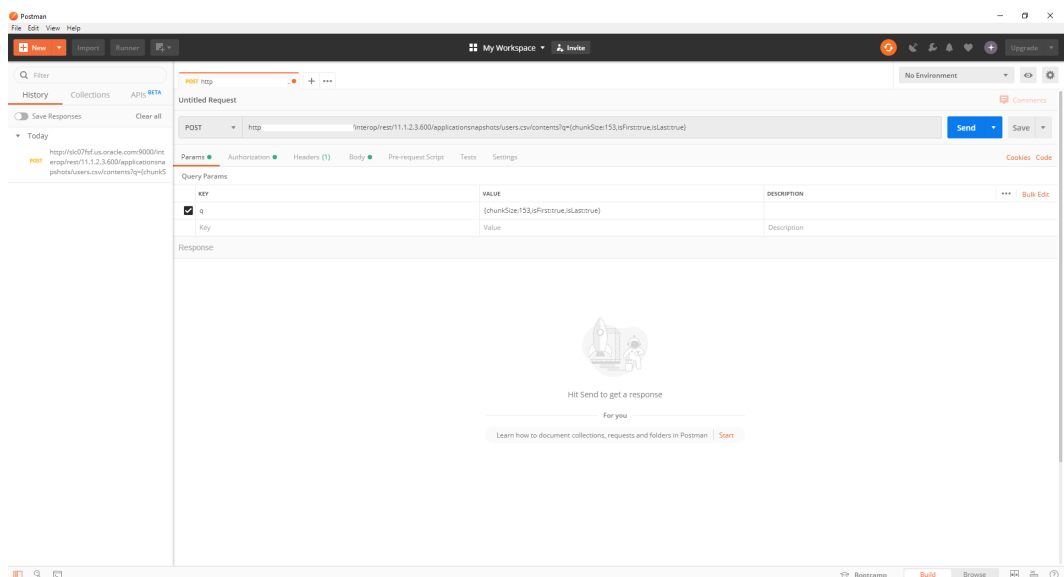
In this example, we upload a file named `users.csv` to our environment, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/`.

For an example of coding parameters, scroll down to the end of this topic.

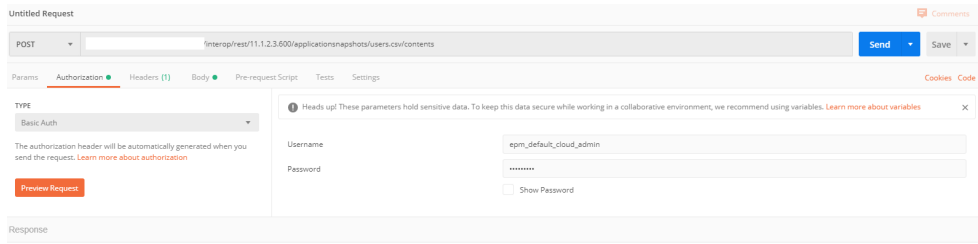
Notes:

- This example uses the 11.1.2.3.600 Upload API, which is a simpler non-chunked version.
- The name of the file is passed in the URL itself, for example, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/applicationsnapshots/users.csv/contents`
- If the filename contains special characters or has whitespace, it should be encoded using any online resource, such as urlencode.org.

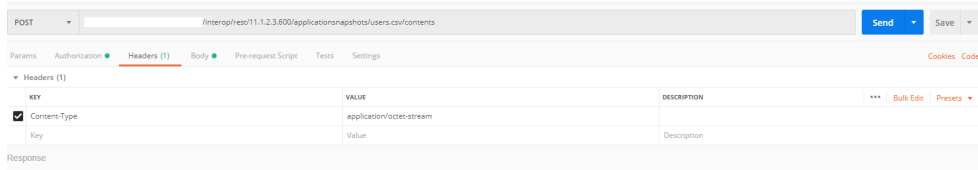
1. Example of Upload parameters.



2. Example of Upload authorization.



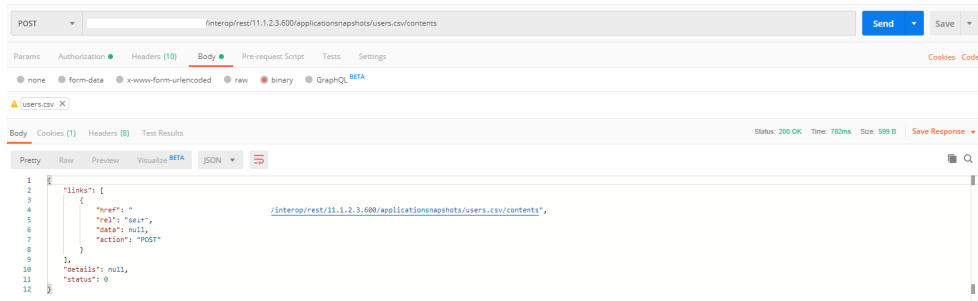
3. Example of Upload headers.



4. Example of Upload body.



5. Example of Upload response on success.



Example: Using REST APIs to Upload to an External Directory with Postman

In this example, we upload a file named `data.txt` to the inbox directory in our environment, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/`.

Notes:

- This example uses the 11.1.2.3.600 Upload API, which is a simpler non-chunked version.
- The name of the file is passed in the URL itself, for example, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/applicationsnapshots/data.txt/contents?extDirPath=inbox`
- If the filename contains special characters or has white space, it must be encoded using any online resource, such as urlencode.org. (See an example at the bottom of this topic: [Example: Using REST APIs to Upload with Postman](#).)

1. Example of parameters for Upload to external directory.

POST `/interop/rest/11.1.2.3.600/applicationsnapshots/data.txt/contents?extDirPath=inbox` Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> extDirPath	inbox	
Key	Value	Description

Response

2. Example of authorization for Upload to external directory.

POST `/interop/rest/11.1.2.3.600/applicationsnapshots/data.txt/contents?extDirPath=inbox` Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Settings Cookies Code

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username `_admin`

Password `*****`

Show Password

Response

3. Example of Upload to external directory headers.

POST `/interop/rest/11.1.2.3.600/applicationsnapshots/data.txt/contents?extDirPath=inbox` Send Save

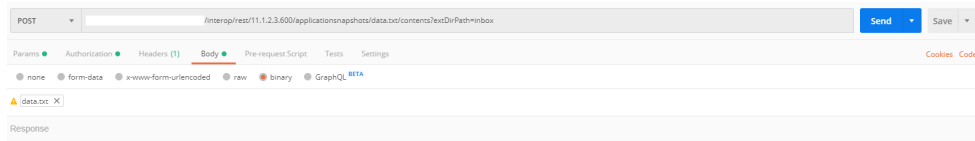
Params Authorization Headers (1) Body Pre-request Script Tests Settings Cookies Code

Headers (1)

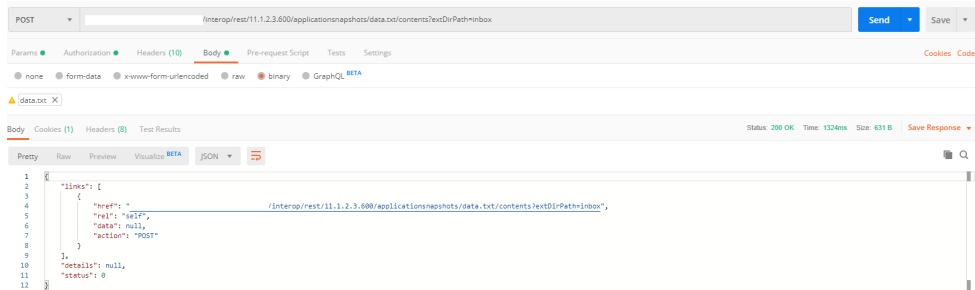
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/octet-stream	
Key	Value	Description

Response

4. Example of Upload to external directory body.



5. Example of Upload to external directory response on success.



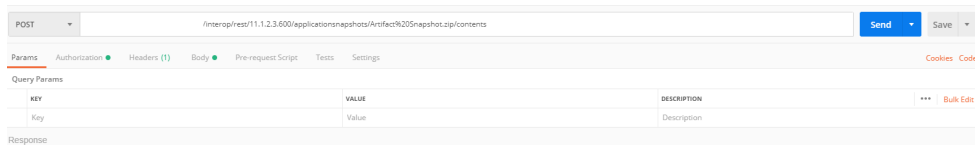
Example: Using REST APIs to Upload a Snapshot with Postman

In this example, we upload a snapshot named `Artifact Snapshot.zip` to our environment, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/`.

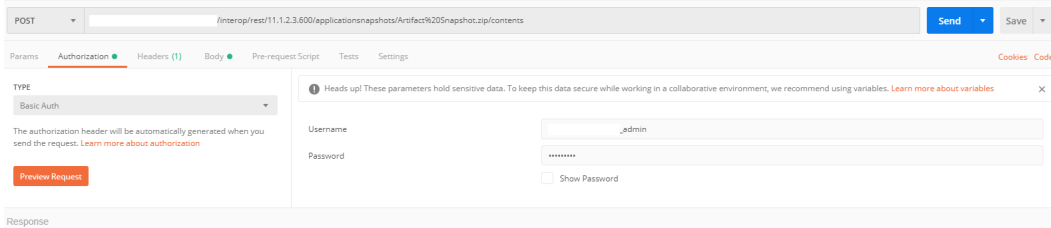
Notes:

- We are using the 11.1.2.3.600 Upload API, which is a simpler non-chunked version.
- The name of the file is passed in the URL itself, for example, `https://<SERVICE_NAME>-<TENANT_NAME>.<SERVICE_TYPE>.<dcX>.oraclecloud.com/interop/rest/11.1.2.3.600/applicationsnapshots/Artifact%20Snapshot.zip/contents`
- If the filename contains special characters or has white space, it must be encoded using any online resource, such as `urlencode.org`

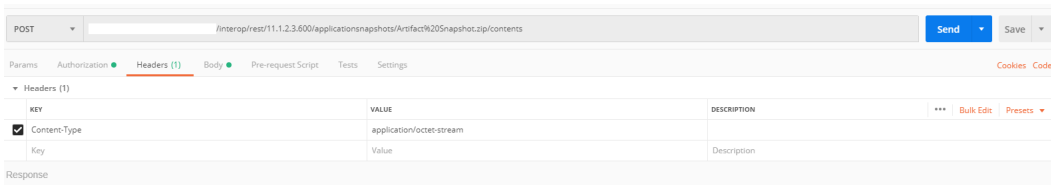
1. Example of Upload snapshot parameters.



2. Example of Upload Snapshot authorization.



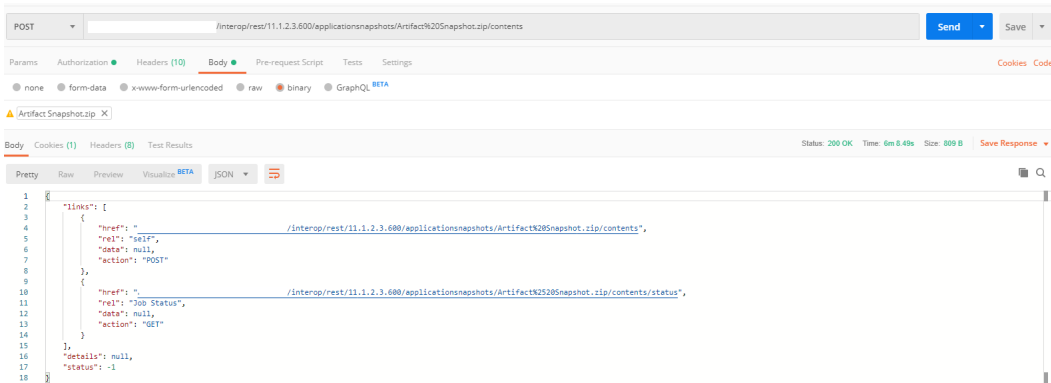
3. Example of Upload Snapshot headers.



4. Example of Upload Snapshot body.



5. Example of Upload Snapshot response on success.



6. Example of checking status for Upload Snapshot .

The screenshot shows the Postman interface for a REST API call. The URL is `/interop/rest/11.1.2.3.600/applicationsnapshots/Artifact%2520Snapshot.zip/contents/status`. The method is GET. The response status is 200 OK, with a time of 9.96s and a size of 504 B. The response body is displayed in JSON format:

```
1 {
2   "links": [
3     {
4       "href": "/interop/rest/11.1.2.3.600/applicationsnapshots/Artifact%2520Snapshot.zip/contents/status",
5       "rel": "self",
6       "data": null,
7       "action": "POST"
8     }
9   ],
10  "details": null,
11  "status": 0,
12  "items": null
13 }
```

G

Profitability and Cost Management Common Helper Functions

Use the Profitability and Cost Management common helper functions as you work with Profitability and Cost Management REST APIs..

Profitability and Cost Management Common Helper Functions for Java

Common Helper Functions for Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import javax.xml.bind.DataConverter;

import org.json.JSONArray;
import org.json.JSONObject;

public class CSSRESTSamples {
    private static String userName;
    private static String password;
    private String serverUrl;
    private String apiVersion;

    public static void main(String[] args) {
        try {
            CSSRESTSamples samples = new CSSRESTSamples("<DOMAINNAME.USERNAME>",
                "<PASSWORD>", "<SERVICE_URL>", "v1");
            // Call sample APIs.
            // samples.addUsers("AddUser2.csv", "test123$", false);
            // samples.removeUsers("test2.csv");
            // samples.assignRole("test3.csv", "Power User");
            // samples.unassignRole("test4.csv", "Viewer");
            // samples.addUsersToGroup("test5.csv", "TestGroup1");
        }
    }
}
```



```

        // samples.removeUsersFromGroup("test6.csv", "TestGroup2");
        //
samples.generateRoleAssignmentReport("JavaSampleReport.csv");
        // samples.generateUserGroupReport("UserGroupReport.csv");
        // samples.addUserToGroups("Group.csv", "user1");
        // samples.removeUserFromGroups("groups.csv", "joe");
        // samples.addGroups("Group1.csv");
        // samples.removeGroups("DeleteGroup1.csv");
        // samples.generateInvalidLoginReport("2021-06-01",
"2021-06-10", "invalidLoginReport141.csv");
        // samples.generateRoleAssignmentAuditReport("2021-06-01",
"2021-06-10", "roleAssignmentaudit_14778.csv");
    } catch (Throwable x) {
        System.err.println("Error: " + x.getMessage());
    }
}

    public CSSRESTSamples(String userName, String password, String
serverUrl, String apiVersion) throws Exception {
        this.userName = userName;
        this.password = password;
        this.serverUrl = serverUrl;
        this.apiVersion = apiVersion;
    }

    public void addUsers(String fileName, String userPassword, boolean
resetPassword) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/users";
            Map<String, String> reqHeaders = new HashMap<String,
String>();
            reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
                .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
String>();
            reqParams.put("filename", fileName);
            reqParams.put("userpassword", userPassword);
            reqParams.put("resetpassword", resetPassword + "");

            Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                "POST");
            String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void removeUsers(String fileName) {

```

```

    try {
        String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
            this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
            HashMap(), url, reqHeaders, reqParams,
            "DELETE");
        String jobStatus =
            CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void assignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
            this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ASSIGN_ROLE");
        reqParams.put("rolename", roleName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
            HashMap(), url, reqHeaders, reqParams,
            "PUT");
        String jobStatus =
            CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void unassignRole(String fileName, String roleName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/users";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
            this.password).getBytes(Charset.defaultCharset())));

```

```

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "UNASSIGN_ROLE");
        reqParams.put("rolename", roleName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

    public void addUsersToGroup(String fileName, String groupName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
            Map<String, String> reqHeaders = new HashMap<String,
String>();
            reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
                .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
String>();
            reqParams.put("filename", fileName);
            reqParams.put("jobtype", "ADD_USERS_TO_GROUP");
            reqParams.put("groupname", groupName);

            Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
                "PUT");
            String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void removeUsersFromGroup(String fileName, String
groupName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
            Map<String, String> reqHeaders = new HashMap<String,
String>();
            reqHeaders.put("Authorization", "Basic " +
DatatypeConverter

```

```

        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset()));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "REMOVE_USERS_FROM_GROUP");
        reqParams.put("groupname", groupName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void addUserToGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);
        reqParams.put("jobtype", "ADD_USER_TO_GROUPS");
        reqParams.put("username", userName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "PUT");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void removeUserFromGroups(String fileName, String userName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
        .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

```

```

    reqParams.put("jobtype", "REMOVE_USER_FROM_GROUPS");
    reqParams.put("username", userName);

    Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
    "PUT");
    String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
    System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void generateRoleAssignmentReport(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentreport";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult =
CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
        "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void generateUserGroupReport(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/usergroupreport";
        Map<String, String> reqHeaders = new HashMap<String,
String>();
        reqHeaders.put("Authorization", "Basic " +
DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String,
String>();
        reqParams.put("filename", fileName);

```

```

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void addGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "POST");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void removeGroups(String fileName) {
    try {
        String url = this.serverUrl + "/interop/rest/security/" +
apiVersion + "/groups";
        Map<String, String> reqHeaders = new HashMap<String, String>();
        reqHeaders.put("Authorization", "Basic " + DatatypeConverter
            .printBase64Binary((this.userName + ":" +
this.password).getBytes(Charset.defaultCharset())));

        Map<String, String> reqParams = new HashMap<String, String>();
        reqParams.put("filename", fileName);

        Map<String, String> restResult = CSSRESTHelper.callRestApi(new
HashMap(), url, reqHeaders, reqParams,
        "DELETE");
        String jobStatus =
CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
        System.out.println(jobStatus);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

    }

    public void generateInvalidLoginReport(String fromDate, String
    toDate,String fileName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/invalidloginreport";
            Map<String, String> reqHeaders = new HashMap<String,
            String>();
            reqHeaders.put("Authorization", "Basic " +
            DatatypeConverter
                .printBase64Binary((this.userName + ":" +
            this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
            String>();
            reqParams.put("from_date", fromDate);
            reqParams.put("to_date", toDate);
            reqParams.put("filename", fileName);

            Map<String, String> restResult =
            CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "POST");
            String jobStatus =
            CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void generateRoleAssignmentAuditReport(String fromDate,
    String toDate,String fileName) {
        try {
            String url = this.serverUrl + "/interop/rest/security/" +
            apiVersion + "/roleassignmentauditreport";
            Map<String, String> reqHeaders = new HashMap<String,
            String>();
            reqHeaders.put("Authorization", "Basic " +
            DatatypeConverter
                .printBase64Binary((this.userName + ":" +
            this.password).getBytes(Charset.defaultCharset())));

            Map<String, String> reqParams = new HashMap<String,
            String>();
            reqParams.put("from_date", fromDate);
            reqParams.put("to_date", toDate);
            reqParams.put("filename", fileName);

            Map<String, String> restResult =
            CSSRESTHelper.callRestApi(new HashMap(), url, reqHeaders, reqParams,
            "POST");
            String jobStatus =
            CSSRESTHelper.getCSSRESTJobCompletionStatus(restResult, reqHeaders);
            System.out.println(jobStatus);
        }
    }
  
```

```

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private static class CSSRESTHelper {
        public static final String REST_CALL_STATUS = "REST_CALL_STATUS";
        public static final String REST_CALL_RESPONSE = "REST_CALL_RESPONSE";

        private static Map<String, String> callRestApi(Map context, String
url, Map<String, String> requestHeaders,
        Map<String, String> requestParams, String methodType) {
            HttpURLConnection urlConnection = null;
            Map<String, String> restResult = new HashMap<String, String>();
            restResult.put(REST_CALL_STATUS, "-1");
            boolean isPostMethod = "POST".equalsIgnoreCase(methodType) ||
"PUT".equalsIgnoreCase(methodType);
            try {
                URI baseUri = new URI(url);
                URI uri = null;
                String reqParams = (requestParams != null ?
buildRequestParams(context, requestParams, isPostMethod)
                : null);
                if (isPostMethod) {
                    uri = new URI(baseUri.getScheme(),
baseUri.getAuthority(), baseUri.getPath(), null, null);
                } else {
                    uri = new URI(baseUri.getScheme(),
baseUri.getAuthority(), baseUri.getPath(), reqParams, null);
                }

                urlConnection = (HttpURLConnection)
uri.toURL().openConnection();
                urlConnection.setRequestMethod(methodType);

                if (requestHeaders != null) {
                    Set<String> requestHeaderKeys = requestHeaders.keySet();
                    for (String requestHeaderKey : requestHeaderKeys) {
                        urlConnection.setRequestProperty(requestHeaderKey,
requestHeaders.get(requestHeaderKey));
                    }
                }

                urlConnection.setUseCaches(false);
                urlConnection.setDoOutput(true);
                urlConnection.setDoInput(true);

                if (isPostMethod) {
                    OutputStreamWriter writer = new
OutputStreamWriter(urlConnection.getOutputStream(),
                        Charset.defaultCharset());
                    writer.write(reqParams);
                    writer.flush();
                }

                if (!isPostMethod) {

```



```

        urlConnection.connect();
    }

    int status = urlConnection.getResponseCode();
    restResult.put(REST_CALL_STATUS,
String.valueOf(status));
    String response = readResponse(context,
        (status >= 400 ?
urlConnection.getErrorStream() : urlConnection.getInputStream()));
    restResult.put(REST_CALL_RESPONSE, response);
} catch (Exception e) {
    restResult.put(REST_CALL_RESPONSE, e.getMessage());
} finally {
    if (urlConnection != null) {
        urlConnection.disconnect();
    }
}
return restResult;
}

private static String buildRequestParams(Map context,
Map<String, String> requestParams, boolean isPostMethod) {
    String reqParams = null;
    try {
        StringBuilder result = new StringBuilder();
        Set<String> reqParamKeys = requestParams.keySet();
        boolean first = true;
        for (String reqParamKey : reqParamKeys) {
            if (first)
                first = false;
            else
                result.append("&");
            String reqParamValue =
requestParams.get(reqParamKey);
            result.append((isPostMethod ?
URLLEncoder.encode(reqParamKey, "UTF-8") : reqParamKey));
            result.append("=");
            result.append((isPostMethod ?
URLLEncoder.encode(reqParamValue, "UTF-8") : reqParamValue));
        }
        reqParams = result.toString();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    return reqParams;
}

private static String readResponse(Map context, InputStream
urlInStream) {
    BufferedReader br = null;
    String response = "";
    try {
        String line;
        br = new BufferedReader(new
InputStreamReader(urlInStream, Charset.defaultCharset()));

```

```

        while ((line = br.readLine()) != null) {
            response += line;
        }
    } catch (Exception e) {
        response += e.getMessage();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return response;
}

private static String getCSSRESTJobUrlFromResponse(String response) {
    String jobUrl = "";
    try {
        JSONObject jsonResponse = new JSONObject(response);
        JSONArray links = (JSONArray) jsonResponse.get("links");
        JSONObject jobStatusLink = (JSONObject) links.get(1);
        jobUrl = jobStatusLink.get("href").toString();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return jobUrl;
}

private static String getCSSRESTJobStatusFromResponse(String
response) {
    String jobStatus = "";
    try {
        JSONObject jsonResponse = new JSONObject(response);
        jobStatus = jsonResponse.get("status").toString();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return jobStatus;
}

private static String getCSSRESTJobCompletionStatus(Map<String,
String> restResult, Map<String, String> reqHeader) {
    String completionStatus = "";
    try {
        String restStatus =
restResult.get(CSSRESTHelper.REST_CALL_STATUS);
        if (restStatus.equalsIgnoreCase("200")) {
            String jobUrl =
getCSSRESTJobUrlFromResponse(restResult.get(CSSRESTHelper.REST_CALL_RESPONSE)
);

            String restJobStatus = "-1";
            Map<String, String> jobStatusResult = null;
            while (restJobStatus.equalsIgnoreCase("-1")) {

```

```

        jobStatusResult =
CSSRESTHelper.callRestApi(new HashMap(), jobUrl, reqHeader, null,
"GET");
        String jobStatusStatus =
jobStatusResult.get(CSSRESTHelper.REST_CALL_STATUS);
        if (jobStatusStatus.equalsIgnoreCase("200")) {
            restJobStatus =
getCSSRESTJobStatusFromResponse(
jobStatusResult.get(CSSRESTHelper.REST_CALL_RESPONSE));
        }
        Thread.sleep(1000);
    }
    completionStatus =
jobStatusResult.get(CSSRESTHelper.REST_CALL_RESPONSE);
    }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return completionStatus;
}
};
}

```

Profitability and Cost Management Common Helper Functions for cURL

Common Helper Functions for cURL

```

#!/bin/sh
#set -x
export PATH=$PATH:<PATH_TO_JQ_BINARY>
SERVER_URL="<SERVICE_URL>"
USERNAME="<USERNAME>"
PASSWORD="<PASSWORD>"
API_VERSION="v1"

# To avoid SSL connection issue in the environment please add -k
option for below curl commands.
funcCallRESTAPI() {
    if [ "$1" == "GET" ] || [ "$1" == "DELETE" ]; then
        if [ "$6" != "" ]; then
            echo `curl -s -u $4:$5 -H "$3" --request $1 -G $2 -d "$6"`
        else
            echo `curl -s -u $4:$5 -H "$3" --request $1 -G $2`
        fi
    else
        if [ "$6" != "" ]; then
            echo `curl -s -u $4:$5 -H "$3" --request $1 $2
-d "$6"`
        else
            echo `curl -s -u $4:$5 -H "$3" --request $1 $2`
        fi
    fi
}

```

```

        fi
    fi
}

funcCSSRESTHelper() {
    jobOutput=$(funcCallRESTAPI "$1" "$2" "$3" "$4" "$5" "$6")
    jobUrl=`echo $jobOutput | jq '.links[1].href'`
    if [ $jobUrl != null ]; then
        jobUrl="${jobUrl%\"}"
        jobUrl="${jobUrl#\"}"
        jobStatus=-1
        while [ $jobStatus == -1 ]; do
            jobOutput=$(funcCallRESTAPI "GET" "$jobUrl"
"$header" "$USERNAME" "$PASSWORD")
            jobStatus=`echo $jobOutput | jq '.status'`
        done
        restStatus=`echo $jobOutput | jq '.details'`
        restStatus="${restStatus%\"}"
        restStatus="${restStatus#\"}"
        statusMessage=""
        if [ $jobStatus == 0 ]; then
            statusMessage="$7 completed successfully."
        #"$restStatus"
        else
            statusMessage=$restStatus
        fi
        echo "$statusMessage"
    else
        failedMessage=`echo $jobOutput | jq '.details'`
        failedMessage="${failedMessage%\"}"
        failedMessage="${failedMessage#\"}"
        echo $failedMessage
    fi
}

funcAddUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&userpassword=$2&resetpassword=$3"
    header="Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
    cssRESTAPI="AddUsers"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveUsers() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="RemoveUsers"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

```

funcAssignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=ASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AssignRole"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcUnassignRole() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/users"
    params="filename=$1&jobtype=UNASSIGN_ROLE&rolename=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="UnassignRole"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAddUsersToGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USERS_TO_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AddUsersToGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveUsersFromGroup() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"

params="filename=$1&jobtype=REMOVE_USERS_FROM_GROUP&groupname=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="RemoveUsersFromGroup"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAddUserToGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=ADD_USER_TO_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-
urlencoded; charset=UTF-8"
    cssRESTAPI="AddUserToGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

```

```

}

funcRemoveUserFromGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1&jobtype=REMOVE_USER_FROM_GROUPS&username=$2"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="RemoveUserFromGroups"
    statusMessage=$(funcCSSRESTHelper "PUT" "$url" "$header" "$USERNAME"
"$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateRoleAssignmentReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateUserGroupReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/usergroupreport"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateUserGroupReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcAddGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="addGroups"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcRemoveGroups() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/groups"
    params="filename=$1"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="removeGroups"
    statusMessage=$(funcCSSRESTHelper "DELETE" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")

```

```

        echo $statusMessage
    }

funcGenerateInvalidLoginReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
invalidloginreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateInvalidLoginReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

funcGenerateRoleAssignmentAuditReport() {
    url="$SERVER_URL/interop/rest/security/$API_VERSION/
roleassignmentauditreport"
    params="from_date=$1&to_date=$2&filename=$3"
    header="Content-Type: application/x-www-form-
urlencoded;charset=UTF-8"
    cssRESTAPI="generateRoleAssignmentAuditReport"
    statusMessage=$(funcCSSRESTHelper "POST" "$url" "$header"
"$USERNAME" "$PASSWORD" "$params" "$cssRESTAPI")
    echo $statusMessage
}

#funcAddUsers test1.csv password false
#funcRemoveUsers test2.csv
#funcAssignRole test3.csv "Power User"
#funcUnAssignRole test4.csv Viewer
#funcAddUsersToGroup test5.csv TestNativeGroup1
#funcRemoveUsersFromGroup test6.csv TestNativeGroup2
#funcGenerateRoleAssignmentReport RoleAssignmentReport.csv
#funcGenerateUserGroupReport UserGroupReport.csv
#funcAddUserToGroups groups.csv joe
#funcRemoveUserFromGroups groups.csv joe
#funcAddGroups CreateGroup1.csv
#funcRemoveGroups DeleteGroup1.csv
#funcGenerateInvalidLoginReport 2021-06-01 2021-06-10
invalidLoginReport.csv
#funcGenerateRoleAssignmentAuditReport 2021-06-01 2021-06-10
roleAssignmentAuditReport.csv

```

Profitability and Cost Management Common Helper Functions for Groovy

Common Helper Functions for Groovy

```

import java.nio.charset.StandardCharsets

import groovy.json.JsonSlurper

```

```

serverUrl="<SERVICE_URL>"
username="<DOMAINNAME.USERNAME>"
password="<PASSWORD>"

apiVersion = "v1";
userCredentials = username + ":" + password;
basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)

def getResponse(is) {
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line+"\n");
    }
    br.close();
    return sb.toString();
}

def getUrlFromResponse(scenario, response, relValue) {
    def object = new JsonSlurper().parseText(response)
    def pingUrlStr
    if (object.status == -1) {
        println "Started - " + scenario
        def links = object.links
        links.each{
            if (it.rel.equals(relValue)) {
                pingUrlStr=it.href
            }
        }
    } else {
        println "Error details: " + object.details
        System.exit(0);
    }
    return pingUrlStr
}

def getJobStatus(pingUrlString, methodType) {

    def pingUrl = new URL(pingUrlString);
    def completed = false;
    while (!completed) {
        pingResponse = executeRequest(pingUrl, methodType, null,
"application/x-www-form-urlencoded");
        status = getJobStatusFromResponse(pingResponse);
        if (status == "Processing") {
            try {
                println "Processing. Please wait..."
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                completed = true
            }
        }
    }
}

```



```

        } else {
            println status
            completed = true
        }
    }
}

def getJobStatusFromResponse(response) {
    def object = new JsonSlurper().parseText(response)
    def status = object.status
    if (status == -1)
        return "Processing"
    else if (status == 0)
        return "Completed"
    else
        return object.details
}

def getJobDetailsFromResponse(response) {
    def object = new JsonSlurper().parseText(response)
    def details = object.details
    if (details != null)
        return object.details
    else
        return null
}

def executeRequest(url, requestType, payload, contentType) {
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
    connection.setDoOutput(true);
    connection.setInstanceFollowRedirects(false);
    connection.setRequestMethod(requestType);
    connection.setRequestProperty("Content-Type", contentType);
    // connection.setRequestProperty("charset",
StandardCharsets.UTF_8);
    connection.setRequestProperty("Authorization", basicAuth);
    connection.setUseCaches(false);

    if (payload != null) {
        OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
        writer.write(payload);
        writer.flush();
    }

    int statusCode
    try {
        statusCode = connection.responseCode;
    } catch (all) {
        println "Error connecting to the URL"
        System.exit(0);
    }

    def response

```

```

    if (statusCode == 200 || statusCode == 201) {
        if (connection.getContentType() != null && !
connection.getContentType().startsWith("application/json")) {
            println "Error occurred in server"
            System.exit(0)
        }
        InputStream is = connection.getInputStream();
        if (is != null)
            response = getResponse(is)
    } else {
        println "Error occurred while executing request"
        println "Response error code : " + statusCode
        InputStream is = connection.getErrorStream();
        if (is != null && connection.getContentType() != null &&
connection.getContentType().startsWith("application/json"))
            println getJobStatusFromResponse(getResponse(is))
        System.exit(0);
    }
    connection.disconnect();
    return response;
}

def addUsersToGroup(fileName, groupName) {

    String scenario = "Adding users in " + fileName + " to group " +
groupName;
    String params = "jobtype=ADD_USERS_TO_GROUP&filename="+ fileName
+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def removeUsersFromGroup(fileName, groupName) {

    String scenario = "Removing users in " + fileName + " from group " +
groupName;
    String params = "jobtype=REMOVE_USERS_FROM_GROUP&filename="+ fileName
+"&groupname="+ groupName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"

```

```

groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def addUserToGroups(fileName, userName) {

    String scenario = "Adding users in " + fileName + " to group " +
userName;
    String params = "jobtype=ADD_USER_TO_GROUPS&filename="+ fileName
+"&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def removeUserFromGroups(fileName, userName) {

    String scenario = "Removing users in " + fileName + " from group "
+ userName;
    String params = "jobtype=REMOVE_USER_FROM_GROUPS&filename="+
fileName + "&username="+ userName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job

```

```

Status"), "GET");
    }
}

def addUsers(fileName, resetPassword, userPassword) {

    String scenario = "Creating users in " + fileName;
    String params = "jobtype=ADD_USERS&filename="+ fileName
+"&resetpassword="+ resetPassword +"&userpassword="+ userPassword;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def addUsers(fileName) {
    addUsers(fileName, null, null);
}

def deleteUsers(fileName) {

    String scenario = "Deleting users in " + fileName;
    String params = null;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
users?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def assignUsersRoles(fileName, roleName) {

    String scenario = "Assigning users in " + fileName + " with role " +
roleName;

```

```

        String params = "jobtype=ASSIGN_ROLE&filename="+ fileName
        +"&rolename="+ roleName;
        def url = null;
        def response = null;
        try {
            url = new URL(serverUrl + "/interop/rest/security/" +
            apiVersion + "/users");
        } catch (MalformedURLException e) {
            println "Please enter a valid URL"
            System.exit(0);
        }
        response = executeRequest(url, "PUT", params, "application/x-www-
        form-urlencoded");
        if (response != null) {
            getJobStatus(getUrlFromResponse(scenario, response, "Job
            Status"), "GET");
        }
    }

def unassignUsersRoles(fileName, roleName) {

    String scenario = "Un-assigning users in " + fileName + " with
    role " + roleName;
    String params = "jobtype=UNASSIGN_ROLE&filename="+ fileName
    +"&rolename="+ roleName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
        apiVersion + "/users");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "PUT", params, "application/x-www-
    form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
        Status"), "GET");
    }
}

def generateRoleAssignmentReport(fileName) {

    String scenario = "Generating Role assignment report in " +
    fileName;
    String params =
    "jobtype=GENERATE_ROLE_ASSIGNMENT_REPORT&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
        apiVersion + "/roleassignmentreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
    }
}

```

```

        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def generateUserGroupReport(fileName) {

    String scenario = "Generating User Group Report in " + fileName;
    String params = "jobtype=GENERATE_USER_GROUP_REPORT&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
usergroupreport");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def addGroups(fileName) {
    println "addgroups"
    String scenario = "Creating Groups in " + fileName;
    String params = "filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" + apiVersion + "/"
groups");
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

def removeGroups(fileName) {

    String scenario = "Deleting Groups in " + fileName;

```

```

String params = null;
def url = null;
def response = null;
try {
    url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/groups?filename=" + fileName);
    } catch (MalformedURLException e) {
        println "Please enter a valid URL"
        System.exit(0);
    }
    response = executeRequest(url, "DELETE", null, "application/x-www-
form-urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
    }
}

def generateRoleAssignmentAuditReport(from_date,to_date,fileName) {

    String scenario = "Generating Role assignment audit report in " +
fileName;
    String params =
"jobtype=GENERATE_ROLE_ASSIGNMENT_AUDIT_REPORT&from_date="+from_date+"&
to_date="+to_date+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/roleassignmentauditreport");
        } catch (MalformedURLException e) {
            println "Please enter a valid URL"
            System.exit(0);
        }
        response = executeRequest(url, "POST", params, "application/x-www-
form-urlencoded");
        if (response != null) {
            getJobStatus(getUrlFromResponse(scenario, response, "Job
Status"), "GET");
        }
    }

def generateInvalidLoginReport(from_date,to_date,fileName) {

    String scenario = "Generating Invalid Login report in " + fileName;
    String params =
"jobtype=GENERATE_INVALID_LOGIN_REPORT&from_date="+from_date+"&to_date=
"+to_date+"&filename="+ fileName;
    def url = null;
    def response = null;
    try {
        url = new URL(serverUrl + "/interop/rest/security/" +
apiVersion + "/invalidloginreport");
        } catch (MalformedURLException e) {
            println "Please enter a valid URL"

```

```

        System.exit(0);
    }
    response = executeRequest(url, "POST", params, "application/x-www-form-
urlencoded");
    if (response != null) {
        getJobStatus(getUrlFromResponse(scenario, response, "Job Status"),
"GET");
    }
}

//Execute commands here
//addUsersToGroup("Users.csv",
"G1");
//PUT
//removeUsersFromGroup("Users.csv",
"G1");
//PUT
//addUsers("AddUsers123.csv", "false",
"newPassword"); //POST
//
addUsers("AddUsers456.csv");

//POST
//
deleteUsers("RemoveUsers.csv"); //DELETE

//assignUsersRoles("Users.csv", "Service
Administrator"); //PUT
//assignUsersRoles("users.csv",
"viewer"); //PUT
//unassignUsersRoles("Users.csv", "Drill
Through"); //PUT
//
generateRoleAssignmentReport("GroovySampleReport3.csv");
// POST
//
generateUserGroupReport("UserGroupReportGroovy.csv");
// POST
//addUserToGroups("Group.csv",
"user1"); //PUT
//removeUserFromGroups("groups.csv",
"joe"); //PUT
//
addGroups("CreateGroup1.csv");
// POST
//
removeGroups("DeleteGroup1.csv");
// DELETE
//generateInvalidLoginReport("2020-06-01", "2021-06-10",
"report12345.csv"); //POST

```


H

Sample Starter Kit for Consultants - Integration with Business Intelligence Cloud Service

This topic describes a sample starter kit that can be used by infrastructure consultants to plan integration for Planning with Business Intelligence Cloud Service.

Prerequisites

- You have accounts for Business Intelligence Cloud Service, Planning, and Oracle Application Express.
- You have considerable technical and functional expertise with Business Intelligence Cloud Service, Planning, Oracle Application Express, REST, Groovy, and scripting.

These are the basic tasks for the sample starter kit for consultants:

- Export data and metadata from Planning using the Planning REST API.
- Download data and metadata to an on-premise server using the Planning REST API.
- Use the metadata to create schema/tables in Business Intelligence Cloud Service using the Business Intelligence Cloud Service REST API.
- Populate the tables in Business Intelligence Cloud Service using the data imported from Planning by using the Business Intelligence Cloud Service REST API.

Note:

If you are using DBaaS, the target reporting database can optionally be accessed by standard tools like SQL Developer and Toad.

Note:

The DataSync tool (available from OTN) can also be used to create tables and load and update data in the tables. Data uploads can be scheduled using the DataSync jobs and native scheduler. This approach will work for Database Schema Service and DBaaS used as reporting database for BICS.

These are the basic steps for the sample starter kit for consultants:

1. Install the scripting engine and deploy demo scripts.
2. Use the SQL APEX REST API client to call a client sample that calls a SQL query with a bind variable passed on the URL.

3. Use the Business Intelligence REST API client to provide methods as necessary for your use case.
4. Use the Planning REST API client to provide methods as necessary for your use case.
5. Incorporate helper functions.
6. Integrate Planning with Business Intelligence Cloud Service using a demo script.

Planning, Business Intelligence Cloud Service, and SQL Web REST services expose functions available in EPM Automate, DataSync, and for direct database SQL/PLSQL calls. The REST APIs can be scripted using any language. This appendix describes a sample starter kit to show how this can be implemented using Groovy for demonstration purposes. For information on REST APIs for Business Intelligence Cloud Service and Application Express, see:

- [Business Intelligence REST APIs](#)
- [Application Express REST APIs](#)

Installing the Scripting Engine and Deploying Demo Scripts

For reference, demo scripts are described here: [Integration of Planning to Business Intelligence Cloud Service](#).

1. Install the Groovy engine, <http://www.groovy-lang.org/install.html>
Select the binary release (<https://bintray.com/artifact/download/groovy/maven/apache-groovy-binary-2.4.5.zip>).
2. Create the files as shown in [Integration of Planning to Business Intelligence Cloud Service](#), and put them in a folder structure similar to the following:

```
pbcsbics
  PBCSBICSAutomation.properties
  com
    oracle
    ceal
    <groovy files>
```

3. Open a shell:

```
cd <yourrootfolder>\pbcsbics
```

On a single line, type:

```
<yourrootfolder>\apache-groovy-binary-2.4.5\groovy-2.4.5\bin\groovy
-classpath
<yourrootfolder>\pbcsbics
<yourrootfolder>\pbcsbics\com\oracle\ceal\PBCSBICSIntegration.groovy
```

SQL Application Express REST API client

The Application Express REST API client sample demonstrates calling an SQL query with a bind variable passed on the URL.

- Client creation using `com.oracle.ceal.ApexRestClient`

```
apexClient=new ApexRestClient(apexRestUrl, proxy Host, proxy Port,cloud
identityDomain, cloud username, cloud password,
ignoreSSLCertificationPathErrors)
```

- Apex REST URL in the format: `https://server`

Example: `https://<SERVER>.oraclecloud.com/apex`

- proxy host:

- * Leave empty if not using a proxy
- * If using a tool like Fiddler for HTTP captures, specify `localhost`.
- * If you need to go through a proxy to connect to Oracle cloud services, specify the proxy host.

- proxy port:

- * Leave empty if not using a proxy
- * If using Fiddler, use `8888`.
- * Otherwise, enter your proxy port.

- Cloud identity domain: this is provided with your cloud login. You can also find this in the APEX URL.

- `ignoreSSLCertificationPathErrors` (true or false): Set this to true if connecting through a proxy like Fiddler.

- Calling an SQL Query defined in APEX / SQL Workshop / RESTful web services

The REST web service must be created first. For more information, read this Oracle By Example: http://www.oracle.com/webfolder/technetwork/tutorials/obe/cloud/13_2/dbservice/restfulws/restfulws.html

```
apexClient.launchSQLQueryUsingGETAndVariableOnUrl("<module name>/<uri>",
"<bind variable>")
```

Example:

```
apexClient.launchSQLQueryUsingGETAndVariableOnUrl("bics/test", "7839")
```

This example uses the following definition of the REST service in APEX:

RESTful Service Module: `bics/` URI Template: `test/{ID}` Method: `GET` Source Type: `Query`
Format: `JSON` Requires Secure Access: `YES` Source: `Select`
`EMPNO,ENAME, JOB,MGR, HIREDATE, SAL, COMM, DEPTNO from EMP where EMPNO = :ID`

The URL call will be in the following format:

```
https://<SERVER>.oraclecloudapps.com/apex/bics/test/7839
```

The response will be in the following format:

Response Content-Type:application/json

```
{ "next": { "$ref": "https://<SERVER>.oraclecloudapps.com/apex/bics/test/7839?page=1" }, "items": [ { "empno": 7839, "ename": "KING", "job": "PRESIDENT", "hiredate": "1981-11-17T00:00:00Z", "sal": 5000, "deptno": 10 } ] }
```

- Calling a PL/SQL defined in APEX / SQL Workshop / RESTful web services

A method is available, but all the configuration work must be done in APEX.

```
apexClient.launchProcUsingGET("<module name>/<uri>")
```

The definition of the REST service in APEX is for this example:

RESTful Service Module: bics/ **URI Template:** plsqli/ **Method:** GET **Source Type:** PL/SQL **Requires Secure Access:** YES **Source:**

```
DECLARE
    prevdeptno    number;
    deptloc       varchar2(30);
    deptname      varchar2(30);
    CURSOR getemps IS select * from emp
                        where ((select job from emp where ename
= :empname) IN ('PRESIDENT', 'MANAGER'))
                        or deptno = (select deptno from emp where
ename = :empname)
                        order by deptno, ename;
BEGIN
    sys.htp.htmlopen;
    sys.htp.headopen;
    sys.htp.title('Departments');
    sys.htp.headclose;
    sys.htp.bodyopen;
    for empregs in getemps
    loop
        if empregs.deptno != prevdeptno or prevdeptno is null
then
            select dname, loc into deptname,
deptloc
            from dept where deptno = (select
deptno from emp where ename = empregs.ename);
            if prevdeptno is not null then
                sys.htp.print('</ul>');
            end if;
            sys.htp.print('Department ' || deptname || '
located in ' || deptloc || '<p/>');
            sys.htp.print('<ul>');
            end if;
            sys.htp.print('<li>' || empregs.ename || ', ' ||
empregs.job || ', ' || empregs.sal || '</li>');
            prevdeptno := empregs.deptno;
        end loop;
    sys.htp.print('</ul>');
    sys.htp.bodyclose;
    sys.htp.htmlclose;
END;
```

URL call will be in the form: `https://<SERVER>.oraclecloudapps.com/apex/bics/plsql/`
Response will be in following format for this specific plsql example

```
Response Content-Type:text/html; charset=UTF-8
<HTML>
<HEAD>
<TITLE>Departments</TITLE>
</HEAD>
<BODY>
</ul>
</BODY>
</HTML>
```

Business Intelligence REST API Client

The Business Intelligence REST API client provides the following methods:

- Client creation using `com.oracle.ceal.BicsRestClient`

```
(BicsClientRestClient.groovy)bicsClient=new BicsRestClient(bics Rest Url,
proxy Host, proxy Port,cloud identityDomain, cloud username, cloud
password, ignoreSSLCertificationPathErrors)
```

- Business Intelligence REST URL in the format: `https://servername`

Example: `https://<SERVER>.oraclecloud.com`

- proxy host:

- * Leave empty if not using a proxy
- * If using a tool like Fiddler for HTTP captures, specify `localhost`.
- * If you need to go through a proxy to connect to Oracle cloud services, specify the proxy host.

- proxy port:

- * Leave empty if not using a proxy
- * If using Fiddler, use `8888`.
- * Otherwise, enter your proxy port.

- Cloud identity domain: this is provided with your cloud login. You can also find this in the BI URL.

- `ignoreSSLCertificationPathErrors` (true or false): Set this to true if connecting through a proxy like Fiddler.

- About bics

```
bicsClient.aboutBics()
```

- List all tables

```
bicsClient.listAllTables()
```

- Get table info

```
bicsClient.getTableInfo(table name)
```

- **Delete table**

```
bicsClient.deleteTable(table name)
```

- **Create a table with X columns and a specific column name prefix**

```
bicsClient.createTableToLoadCSV(table name, number of columns , column prefix)
```

Example:

```
bicsClient.createTableToLoadCSV("ceal_4", 3 , "MYCOL")
```

This creates a table called CEAL_4 with three columns named: MYCOL1, MYCOL2, MYCOL3

By default the columns have the following properties:

```
"dataType":"VARCHAR" // creates a VARCHAR2 column in database
  "length":300,
  "precision":0,
  "nullable":true,
  "defaultValue":null,
```

These values can be modified in `BicsRestClient.groovy` in the `createTableToLoadCSV` method

- **Delete data from table**

```
bicsClient.deleteDataFromTable(table name)
```

- **Load data in table**

```
loadDataInTableUsingCSV(tableName, localCsvFilePath, localCsvFileName,
delimiterInCsv,numberOfColumnsInCsv,numberOfLinesToSkip,columnPrefixIn
Table,isZipped)
```

Example:

```
bicsClient.loadDataInTableUsingCSV("ceal_4","d:\
\temp","export.csv","",3,0,"MYCOL",false)
```

- **Create a table with a specific column name**

```
bicsClient.createTableToLoadCSVWithHeaderNames("ceal_8", listHeaders )
```

- **Load data in table using mappings to specific column names**

```
loadDataInTableUsingCSVAndHeader(tableName, localCsvFilePath,
localCsvFileName,
delimiterInCsv,numberOfLinesToSkip,listHeaders,isZipped)
```

Example:

```
bicsClient.loadDataInTableUsingCSVAndHeader("ceal_8","d:\
\temp",fileNameInZip","",1,listHeaders,false)
```

Planning REST API Client

The Planning REST API client provides the following methods:

- **Client creation using** `com.oracle.ceal.PbcsRestClient`
(`PbcsClientRestClient.groovy`)

```
PbcsRestClient pbcsClient=new
PbcsRestClient(pbcParams.planningRestUrl,pbcParams.interopRestUrl,pbcPa
rams.proxyHost,pbcParams.proxyPort,pbcParams.identityDomain,pbcParams.u
sername, pbcParams.password, pbcParams.ignoreSSLCertificationPathErrors)
```

- **List all files**

```
pbcsClient.listFiles()
```

- **Delete a file**

```
pbcsClient.deleteFile(fileName)
```

- **Export data**

```
response=pbcsClient.exportData(appName, Job name for export, export filename
on server)
```

- **Get job status**

```
pbcsClient.getJobStatus(appName,jobId)
```

- **Download file**

```
pbcsClient.downloadFile(server file name, local destination folder)
```

- **Export metadata**

```
pbcsClient.exportMetaData(appName,job name for metadata export, Export
filename on server)
```

- **Execute LCM Export**

```
pbcsClient.executeLCMExport(snapshot name)
```

- **Run business rule**

```
pbcsClient.runBusinessRule(appname,business rule, runtime prompts)
```

Example:

```
pbcsClient.runBusinessRule("Vision","AggOliv",{Period:Q1,Entity:USA})
```

- **Cube refresh**

```
pbcsClient.cubeRefresh(appname,jobname)
```

Example:

```
pbcsClient.cubeRefresh("Vision","RefreshOliv")
```

- **Run plantype map**

```
pbcsClient.runPlanTypeMap(appname, jobname, cleardata true/false)
```

Example:

```
pbcsClient.cubeRefresh("Vision","RefreshOliv")pbcsClient.runPlanTypeMap("Vis
ion","MapOliv","false")
```

- **Run Ruleset**

```
pbcsClient.runRuleSet(appname, ruleset)
```

- **Import data**

```
pbcsClient.importData(appname, jobname, export filename)
```

Last parameter (export filename) **defaults to jobname if no server import filename is specified**

- **Import metadata**

```
pbcsClient.importMetaData(appname, jobname, export metadata filename)
```

Last parameter **defaults to jobname if no server import filename is specified**

Example:

```
pbcsClient.importMetaData("Vision","ImportMetaOliv","ExportMetadataOliv.zip")
```

- **Execute LCM import**

```
pbcsClient.executeLCMImport(snapshot name)
```

- **Upload file**

```
pbcsClient.uploadFile(Local folder containing file to upload to server,file to upload)
```

Example:

```
//local folder, and filename as parameters
pbcsClient.uploadFile("d:\\temp","ExportOliv.zip")
```

Helper Functions

- The properties file containing connection parameters can also be accessed using this class:

```
com.oracle.ceal.BICSAutomationParameters OR
com.oracle.ceal.PBCSAutomationParameters OR
com.oracle.ceal.APEXAutomationParameters
```

The Properties file must contain for BICS:

```
proxyHost=
proxyPort=
ignoreSSLCertificationPathErrors=true or false
bicsRestUrl=https://biserverurl
bicsIdentityDomain=
bicsUsername=
bicsPassword=

// this loads a file named PBCSBICSAutomation.properties
PbcsRestClient pbcsClient

PBCSAutomationParameters pbcsParams=new
PBCSAutomationParameters('PBCSBICSAutomation.properties')
```



```
pbcsClient=new
PbcsRestClient(pbcParams.planningRestUrl,pbcParams.interopRestUrl,pbcParams.proxyHost,pbcParams.proxyPort,pbcParams.identityDomain,pbcParams.username, pbcParams.password, pbcParams.ignoreSSLCertificationPathErrors)
```

Properties file must contain for Planning:

```
pbcsPlanningRestUrl=https://<SERVER>/rest/11.1.2.3.600
pbcsInteropRestUrl=https://<SERVER>/interop/rest/11.1.2.3.600
pbcsIdentityDomain=
pbcsUsername=
pbcsPassword=
proxyHost=
  proxyPort=
ignoreSSLCertificationPathErrors=true or false

// this loads a file named PBCSBICSAutomation.properties
ApexRestClient apexClient

APEXAutomationParameters apexParams=new
APEXAutomationParameters('PBCSBICSAutomation.properties')

apexClient=new
ApexRestClient(apexParams.apexRestUrl,apexParams.proxyHost,apexParams.proxyPort,apexParams.identityDomain,apexParams.username, apexParams.password, apexParams.ignoreSSLCertificationPathErrors)
```

The properties file must contain for Planning:

```
apexRestUrl=https://apexserver/apex
apexIdentityDomain=
apexUsername=
apexPassword=
proxyHost=
proxyPort=
ignoreSSLCertificationPathErrors=true or false
```

- The Planning REST client also contains helper functions for dealing with CSV files:

- Finding the number of columns in a .csv file

Example:

```
nbColsInCsv=pbcsClient.findNbOfColsInCSV("d:\\temp\\", "export.csv", ",")
```

- Finding header names in a .csv file (first line):

```
listHeaders=pbcsClient.getHeadersInCSVAsList(folder containing csv file, csv filename, delimiter)
```

Example:

```
listHeaders=pbcsClient.getHeadersInCSVAsList("d:\\temp\\", fileName, ",")
```

- The Planning REST client also contains helpers functions for dealing with asynchronous calls:

```
Class WaitForCode
    Method retry(sleep time, nb of retries) { code to
run }
```

Example:

```
// Looping to get jobId status while it s being processed on server
// In this example waiting 6 secs each time, and trying 100
times to get a valid status (not processing)
WaitForCode.retry(6000,100){
// second parameter is jobId (is obtained from json response
from server)
    def responseJobStatus
        responseJobStatus=pbcClient.getJobStatus("Vision",jobId)
if (responseJobStatus.contains("Processing")) throw new
Exception("Job not finished")
}
```

- The Business Intelligence Cloud Service REST client also contains helpers functions for trimming lists:

```
def truncateList(listName, truncateLength)
```

Example to truncate headers for columns to 30 characters:

```
def listHeaders
listHeaders=pbcClient.getHeadersInCSVAsList("d:\\temp\\",
fileNameInZip, ",")
listHeaders=bicsClient.truncateList(listHeaders, 30)
```

Integration of Planning to Business Intelligence Cloud Service

The demo scripts in the following topics show Groovy examples of integration for Planning and Business Intelligence Cloud Service.

Related Topics

- [Groovy Sample – PBCSBICSIntegration.groovy](#)
- [Groovy Sample – PbcRestClient.groovy](#)
- [Groovy Sample – PbcRestClient.groovy](#)
- [Groovy Sample – BicsRestClient.groovy](#)
- [Groovy Sample – ApexRestClient.groovy](#)

Groovy Sample – PBCSBICSIntegration.groovy

```

package com.oracle.ceal

class PBCSBICSIntegration {
    static main(args) {
        def pbcsexportfiles

        PbcRestClient pbcClient
        BicsRestClient bicsClient
        ApexRestClient apexClient

        PBCSAutomationParameters pbcParams=new
        PBCSAutomationParameters('PBCSBICSAutomation.properties')
        if (pbcParams.isConfigValid() == true) {
            pbcClient=new
            PbcRestClient(pbcParams.planningRestUrl,pbcParams.interopRestUrl,pbcParams.proxyHost,pbcParams.proxyPort,pbcParams.identityDomain,pbcParams.username, pbcParams.password, pbcParams.ignoreSSLCertificationPathErrors)

            pbcClient.listFiles()

            pbcClient.deleteFile("ExportOliv.zip")
            pbcClient.deleteFile("ExportMetadataOliv.zip")

            def response
            //last parameter is server filename for export. If not set, this defaults to jobname as filename

            response=pbcClient.exportData("Vision","JobOliv","ExportOliv.zip")
            String jobId = "";
            jobId=pbcClient.getJobIdFromJSONResponse(response)
            if (jobId!="") println "Export running with jobId:"+jobId

            // Looping to get jobId status while it s being processed on
            server
            // In this example waiting 6 secs each time, and trying 100
            times to get a valid status (not processing)
            WaitForCode.retry(6000,100){
                // second parameter is jobid (is obtained from json response
            from server)
                def responseJobStatus
                responseJobStatus=pbcClient.getJobStatus("Vision",jobId)
                if (responseJobStatus.contains("Processing")) throw new
            Exception("Job not finished"
            }

            //download server file name to local folder
            pbcClient.downloadFile("ExportOliv.zip","d:\\temp")

            //last parameter is server filename for export. If not set, this defaults to jobname as filename

```

```

pbcsClient.exportMetaData("Vision","JobOlivMeta","ExportMetadadataOliv.zip")

pbcsClient.downloadFile("ExportMetadadataOliv.zip","d:\
\temp")

pbcsExportfiles=pbcsClient.unZip("d:\\temp\
\ExportOliv.zip","d:\\temp\\")
pbcsExportfiles.each { fileNameInZip ->
    println "-->" + fileNameInZip
    println "Nb of cols in
csv:" + pbcsClient.findNbOfColsInCSV("d:\\temp\\", fileNameInZip, ",")
    def headers
    headers=pbcsClient.getHeadersInCSVAsList("d:\\temp\\",
fileNameInZip, ",")
    headers.each { header ->
        println "header --" + header + "--"
    }
    println "<--"
}

} else {
    println "Configuration for PBCS is invalid. Please check
PBCSBICSAutomation.properties"
}

BICSAutomationParameters bicsParams=new
BICSAutomationParameters('PBCSBICSAutomation.properties')
if (bicsParams.isConfigValid() == true) {

    // load to bics

    bicsClient=new
BicsRestClient(bicsParams.bicsRestUrl,bicsParams.proxyHost,bicsParams.p
roxyPort,bicsParams.identityDomain,bicsParams.username,
bicsParams.password, bicsParams.ignoreSSLCertificationPathErrors)
    bicsClient.aboutBics()
    bicsClient.listAllTables()
    bicsClient.getTableInfo("ceal_4")
    bicsClient.deleteTable("ceal_4")
    // this creates a table with x columns MYCOL1 MYCOL2 MYCOL3
    bicsClient.createTableToLoadCSV("ceal_4", 3, "MYCOL")
    bicsClient.deleteDataFromTable("ceal_4")
    //loadDataInTableUsingCSV(tableName, localCsvFilePath,
localCsvFileName,
delimiterInCsv,numberofColumnsInCsv,numberofLinesToSkip,columnPrefixInT
able,isZipped)
    bicsClient.loadDataInTableUsingCSV("ceal_4", "d:\
\temp","export.csv",",",3,0,"MYCOL",false)

    println "**Uploading each file from zip**"
    pbcsExportfiles.each { fileNameInZip ->
        println "-->" + fileNameInZip
        def nbColsInCsv
        nbColsInCsv=pbcsClient.findNbOfColsInCSV("d:\\temp\\",

```



```

pbcsUsername=
pbcsPassword=
proxyHost=
proxyPort=
ignoreSSLCertificationPathErrors=false
bicsRestUrl=https://bicsserver
bicsIdentityDomain=
bicsUsername=
bicsPassword=
apexRestUrl=https://dbserver/apex
apexIdentityDomain=
apexUsername=
apexPassword=

```

Groovy Sample – PbcsRestClient.groovy

```

package com.oracle.ceal

import javax.net.ssl.HostnameVerifier
import javax.net.ssl.HttpURLConnection
import javax.net.ssl.SSLContext
import javax.net.ssl.SSLSession
import javax.net.ssl.TrustManager
import javax.net.ssl.X509TrustManager

import java.net.HttpURLConnection

import java.util.regex.Pattern
import java.util.regex.Matcher
import java.util.zip.ZipEntry
import java.util.zip.ZipFile

class PbcsRestClient {
    private HttpURLConnection connection
    private def planningUrl
    private def interopUrl
    private def proxyHost
    private def proxyPort
    private def user
    private def pwd
    private def domain
    private def ignoreSSLCertsErrors

    public PbcsRestClient(planningServerUrl,
interopServerUrl,httpProxyHost, httpProxyPort,
identityDomain,username, password, ignoreSSLCertificationPathErrors) {
        planningUrl=planningServerUrl
        interopUrl=interopServerUrl
        proxyHost=httpProxyHost
        proxyPort=httpProxyPort
        domain=identityDomain
        user=username
        pwd=password
        ignoreSSLCertsErrors=ignoreSSLCertificationPathErrors
    }
}

```

```

    }

    def setProxyParams() {
        Properties systemProperties = System.getProperties()
        systemProperties.setProperty("http.proxyHost", proxyHost)
        systemProperties.setProperty("http.proxyPort", proxyPort)
        systemProperties.setProperty("https.proxyHost", proxyHost)
        systemProperties.setProperty("https.proxyPort", proxyPort)
    }

    def setSSLParams() {
        if (ignoreSSLCertsErrors != null &&
            ignoreSSLCertsErrors.toUpperCase() == "TRUE") {
            println "Ignoring SSL certification path errors"
            // Disable SSL cert validation

            def hostnameVerifier = [
                verify: { hostname, session -> true }
            ]
            def trustManager = [
                checkServerTrusted: { chain, authType -> },
                checkClientTrusted: { chain, authType -> },
                getAcceptedIssuers: { null }
            ]

            HttpsURLConnection.setDefaultHostnameVerifier(hostnameVerifier
                as HostnameVerifier)

            HttpsURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory())

            SSLContext context = SSLContext.getInstance("SSL")
            context.init(null, [trustManager as X509TrustManager] as
                TrustManager[], null)
        }
    }

    def openConnection(restUrl, method, localFileNameWithPathForStorage) {
        println "Opening connection to $restUrl with method:$method"
        int statusCode

        setProxyParams()
        setSSLParams()

        URL newUrl
        newUrl = new URL(restUrl)

        connection = (HttpURLConnection) newUrl.openConnection()

        connection.setDoOutput(true)
        connection.setDoInput(true)
        connection.setUseCaches(false)
        if (method == "")
    }

```

```

        connection.setRequestMethod("GET")
    else
        connection.setRequestMethod(method)

    connection.setRequestProperty("Content-Type","application/x-
www-form-urlencoded")

    String userCredentials = domain + "." + user + ":" + pwd
    String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getB
ytes())
    connection.setRequestProperty("Authorization", basicAuth)

    String response=""
    try {
        statusCode = connection.responseCode
        println "Connection status code: $statusCode "
        if (statusCode==401) {
            println "Not authorized"
        }
        if (statusCode==200) {
            println "Authentication succeeded"
            println "Server response:"
            println "-----"

response=displayServerResponse(connection,localFileNameWithPathForStora
ge)

            println "-----"
        }
        if (statusCode==400) {
            println "Bad request"
            println "Server response:"
            println "-----"
            response=displayServerResponse(connection,"")
            println "-----"
        }
    } catch (Exception e) {
        println "Error connecting to the URL"
        println e.getMessage()
    } finally {
        if (connection != null) {
            connection.disconnect();
        }
    }

    return response
}

def
displayServerResponse(connection,localFileNameWithPathForStorage) {
    InputStream is;
    if (connection.getResponseCode()==200) {
        is=connection.getInputStream();
    } else {
        is=connection.getErrorStream();
    }
}

```



```

    }
    println "Response Content-Type:"+connection.getContentType()
    if (connection.getContentType().contains("application/json")) {
        BufferedReader br = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = br.readLine()) != null) {
            sb.append(line+"\n");
        }
        br.close();
        println sb
        return sb.toString()
    } else {
        if (connection.getResponseCode()==200) {
            //storing content
            final int BUFFER_SIZE = 5 * 1024 * 1024;
            def fileExt = connection.getHeaderField("fileExtension");
            println "Downloading file with fileExtension header:"+fileExt
            if (fileExt!=null) {
                def saveFilePath = localFileNameWithPathForStorage;
                File f = new File(saveFilePath);
                is = connection.getInputStream();
                FileOutputStream outputStream = new FileOutputStream(f);
                int bytesRead = -1;
                byte[] buffer = new byte[BUFFER_SIZE];
                while ((bytesRead = is.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }
                println "Downloaded file to $localFileNameWithPathForStorage"
                return localFileNameWithPathForStorage
            } else {
                println "Could not find fileExtension header"
            }
        }
    }
}

return ""
}

def listFiles() {
    println "***Listing files**"
    def restUrl=interopUrl+"/applicationsnapshots"
    def response

    response=openConnection(restUrl,"GET","")
    println "*****"
}

def deleteFile(serverFileName) {
    println "***deleting file**"
    def restUrl=interopUrl+"/applicationsnapshots/"+serverFileName
    def response
}

```

```

        response=openConnection(restUrl,"DELETE","")
        println "****"
    }

    def getJobStatus(appName,jobId) {
        println "***get Job status**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs/" +
jobId

        def response
        response=openConnection(restUrl,"GET","")
        println "****"
        return response
    }

    def exportData(appName,jobName, exportServerFileName) {
        println "***Exporting data**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=EXPORT_DATA"
        if (exportServerFileName!="") {
            def exportFileJSON="{exportFileName:$exportServerFileName}"
            restUrl=restUrl+"&parameters=" + exportFileJSON
        }

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
        return response
    }

    def getJobIdFromJSONResponse(response) {
        def jobId=""
        try {
            Pattern regex = Pattern.compile("\"jobId\":\\d+");
            Matcher matcher = regex.matcher(response);
            while (matcher.find()) {
                jobId = matcher.group(0).replace("\"jobId\":\"","");
            }
        } catch (Exception e) {
            println "No jobId found in server response"
        }
        return jobId
    }

    def downloadFile(serverFileName,localFolderForStorage) {
        println "***Downloading file**"
        def restUrl=interopUrl+"/
applicationsnapshots/"+serverFileName+ "/contents"
        def response

        response=openConnection(restUrl,"GET",localFolderForStorage+"/"+serverF
ileName)
        println "****"
    }

```

```

def exportMetaData(appName,jobName, exportServerFileName) {
  println "***Exporting metadata***"
  def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=EXPORT_METADATA"
  if (exportServerFileName!="") {
    def exportFileJSON="{exportZipFileName:$exportServerFileName}"
    restUrl=restUrl+"&parameters=" + exportFileJSON
  }

  def response
  response=openConnection(restUrl,"POST","")
  println "*****"
}

def executeLCMExport(snapshotName) {
  println "***Exporting snapshot***"
  def typeExport="{type:export}"
  def restUrl=interopUrl+"/applicationsnapshots/"+snapshotName+ "/
migration?q="+typeExport
  def response
  response=openConnection(restUrl,"POST","")
  println "*****"
}

def executeLCMImport(snapshotName) {
  println "***Importing snapshot***"
  def typeImport="{type:import}"
  def restUrl=interopUrl+"/applicationsnapshots/"+snapshotName+ "/
migration?q="+typeImport
  def response
  response=openConnection(restUrl,"POST","")
  println "*****"
}

def runBusinessRule(appName,jobName, JSONRuntimePrompt) {
  println "***Running business rule***"
  def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=RULES"
  if (JSONRuntimePrompt!="") {
    // Example for JSONRuntimePrompt {Period:Q1,Entity:USA}
    restUrl=restUrl+"&parameters=" + JSONRuntimePrompt
  }

  def response
  response=openConnection(restUrl,"POST","")
  println "*****"
}

def runRuleSet(appName,jobName) {
  println "***Running rule set***"
  def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=RULESET"
  def response
  response=openConnection(restUrl,"POST","")
}

```

```

        println "****"
    }

    def cubeRefresh(appName,jobName) {
        println "***Refreshing cube**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=CUBE_REFRESH"

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def runPlanTypeMap(appName,jobName, clearData) {
        println "***Running map (job of type plan_type_map)**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=PLAN_TYPE_MAP"
        if (clearData!=null && clearData.toUpperCase()=="FALSE") {
            restUrl=restUrl+"&parameters={clearData:false}"
        } else {
            println "Clear data is set to true (default)"
        }
        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def importData(appName,jobName, importFileName) {
        println "***Importing data**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=IMPORT_DATA"
        if (importFileName!="") {
            def exportFileJSON="{importFileName:$importFileName}"
            restUrl=restUrl+"&parameters=" + exportFileJSON
        }

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def importMetaData(appName,jobName, importZipFileName) {
        println "***Importing metadata**"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=IMPORT_METADATA"
        if (importZipFileName!="") {
            def exportFileJSON="{importZipFileName:$importZipFileName}"
            restUrl=restUrl+"&parameters=" + exportFileJSON
        }

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }
}

```

```

def uploadFile(localPath,fileName) {
println "***Uploading file**"
def restUrl=interopUrl+"/applicationsnapshots/"+fileName

final int DEFAULT_CHUNK_SIZE = 50 * 1024 * 1024;
int packetNo = 1;
boolean status = true;
byte[] lastChunk = null;
File f = new File(localPath+"/"+fileName);
InputStream fis = null;
long totalFileSize = f.length();
boolean isLast = false;
Boolean isFirst = true;
boolean firstRetry = true;
int lastPacketNo = (int) (Math.ceil(totalFileSize/ (double)
DEFAULT_CHUNK_SIZE));
long totalbytesRead = 0;
try {
fis = new BufferedInputStream(new
FileInputStream(localPath+"/"+fileName));
while (totalbytesRead < totalFileSize && status) {
int nextChunkSize = (int) Math.min(DEFAULT_CHUNK_SIZE,
totalFileSize - totalbytesRead);
if (lastChunk == null) {
lastChunk = new byte[nextChunkSize];
int bytesRead = fis.read(lastChunk);
totalbytesRead += bytesRead;
if (packetNo == lastPacketNo) {
isLast = true;
}
status = sendRequestToRestForUpload(restUrl,isFirst,
isLast,lastChunk);
isFirst=false;
if (status) {
println "\r" + ((100 * totalbytesRead)/
totalFileSize) + "% completed";
} else {
break;
}
packetNo = packetNo + 1;
lastChunk = null;
}
}
} catch (Exception e) {
println "Exception occurred while uploading file";
println e.getMessage()
} finally {
if (null != fis) {
}
}
println "****"
}

def sendRequestToRestForUpload(restUrl,isFirst, isLast,lastChunk) {

```

```

        def url=restUrl+"/contents?
q={isLast:$isLast,chunkSize:"+lastChunk.length+",isFirst:$isLast}"
        println "Opening connection for upload to $url"
        int statusCode

        setProxyParams()
        setSSLParams()

        URL newUrl
        newUrl=new URL(url)

        connection = (HttpURLConnection) newUrl.openConnection()

        connection.setDoOutput(true)
        connection.setDoInput(true)
        connection.setUseCaches(false)
        connection.setRequestMethod("POST")

        connection.setRequestProperty("Content-Type","application/
octet-stream")

        String userCredentials = domain + "." + user + ":" + pwd
        String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes())
        connection.setRequestProperty("Authorization", basicAuth)
        DataOutputStream wr = new
DataOutputStream(connection.getOutputStream());
        wr.write(lastChunk);
        wr.flush();

        boolean status = false
        int execStatus
        try {
            execStatus = connection.getResponseCode();
            InputStream is = connection.getInputStream();
            BufferedReader br = new BufferedReader(new
InputStreamReader(is));
            StringBuilder sb = new StringBuilder();
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line+"\n");
            }
            br.close();
            String stat = sb.toString();
            if (null == stat || stat.isEmpty()) {
                return status;
            } else {
                if (200 == execStatus) {
                    println stat
                }
            }
        } catch (Exception e) {

```

```

        println "Exception occurred while uploading file";
        println e.getMessage()
    } finally {
        if (connection != null) {
            connection.disconnect();
        }
    }
}

// Helper functions

def unzip(fileName, destinationFolder) {
    // code from http://www.oracle.com/technetwork/articles/java/
    compress-1565076.html
    println ("**Unzipping "+fileName+"**")
    def fileList=[]
    int BUFFER = 2048;
    try {
        BufferedOutputStream dest = null;
        BufferedInputStream is = null;
        ZipEntry entry;
        ZipFile zipfile = new ZipFile(fileName);
        Enumeration e = zipfile.entries();
        while(e.hasMoreElements()) {
            entry = (ZipEntry) e.nextElement();
            //println("Extracting: " +entry);
            is = new BufferedInputStream(zipfile.getInputStream(entry));
            int count;
            byte[] data;
            data = new byte[BUFFER];
            FileOutputStream fos = new
FileOutputStream(destinationFolder+"/"+entry.getName());
            fileList.push(entry.getName())
            dest = new BufferedOutputStream(fos, BUFFER);
            while ((count = is.read(data, 0, BUFFER)) != -1) {
                dest.write(data, 0, count);
            }
            dest.flush();
            dest.close();
            is.close();
        }
    } catch (FileNotFoundException fnfe) {
        println "Make sure there is not folder in the zip . Zip not
processed"
        //fnfe.printStackTrace();
    } catch (Exception e) {
        println "An error occurred while unzipping."
        println e.getMessage()
    }
    return fileList
    println "*****"
}

def findNbOfColsInCSV(filePath, fileName, delimiter) {

```

```

File csvFile=new File (filePath+"/"+fileName);
Scanner scanner = new Scanner(csvFile);
scanner.useDelimiter(delimiter);

def nbCols
nbCols=0
if (scanner.hasNextLine()) {
    String[] vals = scanner.nextLine().split(delimiter);
    nbCols=vals.size()
}
scanner.close();

return nbCols
}

def getHeadersInCSVAsList(filePath, fileName, delimiter) {
    String[] headers =[]

    BufferedReader br = new BufferedReader(new
FileReader(filePath+"/"+fileName));
    String firstLine = br .readLine();
    println "First line is : " + firstLine
    println "Removing all non ascii chars from line"
    firstLine = firstLine.replaceAll("[^ -~]", "");
    firstLine = firstLine.replaceAll(" ", "");
    // firstLine = firstLine.replaceAll("\\"", "");
    headers = firstLine.split(delimiter);

    def headersList = headers as List
    headersList = headersList.collect { it.trim() }

    return headersList
}

}

class WaitForCode {

static retry( sleepTime, nbOfRetries, Closure logicToRun){
    Throwable caught = null
    for(int i=0; i<nbOfRetries; i++){
        try {
            return logicToRun.call()
        } catch(Throwable t){
            caught = t
            println ("Retrying...")
            Thread.sleep(sleepTime)
        }
    }
    println ("Retry count limit exceeded. Stopping check.")
    throw caught
}
}
}

```


Groovy Sample – PbcRestClient.groovy

```
package com.oracle.ceal

import javax.net.ssl.HostnameVerifier
import javax.net.ssl.HttpURLConnection
import javax.net.ssl.SSLContext
import javax.net.ssl.SSLSession
import javax.net.ssl.TrustManager
import javax.net.ssl.X509TrustManager

import java.net.HttpURLConnection

import java.util.regex.Pattern
import java.util.regex.Matcher
import java.util.zip.ZipEntry
import java.util.zip.ZipFile

class PbcRestClient {
    private HttpURLConnection connection
    private def planningUrl
    private def interopUrl
    private def proxyHost
    private def proxyPort
    private def user
    private def pwd
    private def domain
    private def ignoreSSLCertsErrors

    public PbcRestClient(planningServerUrl, interopServerUrl, httpProxyHost,
httpProxyPort, identityDomain, username, password,
ignoreSSLCertificationPathErrors) {
        planningUrl=planningServerUrl
        interopUrl=interopServerUrl
        proxyHost=httpProxyHost
        proxyPort=httpProxyPort
        domain=identityDomain
        user=username
        pwd=password
        ignoreSSLCertsErrors=ignoreSSLCertificationPathErrors
    }

    def setProxyParams() {
        Properties systemProperties = System.getProperties()
        systemProperties.setProperty("http.proxyHost", proxyHost)
        systemProperties.setProperty("http.proxyPort", proxyPort)
        systemProperties.setProperty("https.proxyHost", proxyHost)
        systemProperties.setProperty("https.proxyPort", proxyPort)
    }

    def setSSLParams() {
```

```

        if (ignoreSSLCertsErrors !=null &&
ignoreSSLCertsErrors.toUpperCase()=="TRUE") {
            println "Ignoring SSL certification path errors"
            // Disable SSL cert validation

            def hostnameVerifier = [
                verify: { hostname, session -> true }
            ]
            def trustManager = [
                checkServerTrusted: { chain, authType -> },
                checkClientTrusted: { chain, authType -> },
                getAcceptedIssuers: { null }
            ]

            HttpURLConnection.setDefaultHostnameVerifier(hostnameVerifier as
HostnameVerifier)

            HttpURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory(
))

                SSLContext context = SSLContext.getInstance("SSL")
                context.init(null, [trustManager as X509TrustManager] as
TrustManager[], null)

            }
        }

        def openConnection(restUrl,method,localFileNameWithPathForStorage)
{
            println "Opening connection to $restUrl with method:$method"
            int statusCode

            setProxyParams()
            setSSLParams()

            URL newUrl
            newUrl=new URL(restUrl)

            connection = (HttpURLConnection) newUrl.openConnection()

            connection.setDoOutput(true)
            connection.setDoInput(true)
            connection.setUseCaches(false)
            if (method=="")
                connection.setRequestMethod("GET")
            else
                connection.setRequestMethod(method)

            connection.setRequestProperty("Content-Type","application/x-
www-form-urlencoded")

            String userCredentials = domain + "." +user + ":" + pwd
            String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getB

```

```

ytes())
    connection.setRequestProperty("Authorization", basicAuth)

    String response=""
    try {
        statusCode = connection.responseCode
        println "Connection status code: $statusCode "
        if (statusCode==401) {
            println "Not authorized"
        }
        if (statusCode==200) {
            println "Authentication succeeded"
            println "Server response:"
            println "-----"

response=displayServerResponse(connection,localFileNameWithPathForStorage)
            println "-----"
        }
        if (statusCode==400) {
            println "Bad request"
            println "Server response:"
            println "-----"
            response=displayServerResponse(connection,"")
            println "-----"
        }
    } catch (Exception e) {
        println "Error connecting to the URL"
        println e.getMessage()
    } finally {
        if (connection != null) {
            connection.disconnect();
        }
    }

    return response
}

def displayServerResponse(connection,localFileNameWithPathForStorage) {
    InputStream is;
    if (connection.getResponseCode()==200) {
        is=connection.getInputStream();
    } else {
        is=connection.getErrorStream();
    }
    println "Response Content-Type:"+connection.getContentType()
    if (connection.getContentType().contains("application/json")) {
        BufferedReader br = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = br.readLine()) != null) {
            sb.append(line+"\n");
        }
        br.close();
        println sb
    }
}

```

```

        return sb.toString()
    } else {
    if (connection.getResponseCode()==200) {
    //storing content
        final int BUFFER_SIZE = 5 * 1024 * 1024;
        def fileExt = connection.getHeaderField("fileExtension");
        println "Downloading file with fileExtension
header:"+fileExt
        if (fileExt!=null) {
            def saveFilePath = localFileNameWithPathForStorage;
            File f = new File(saveFilePath);
            is = connection.getInputStream();
            FileOutputStream outputStream = new
FileOutputStream(f);
            int bytesRead = -1;
            byte[] buffer = new byte[BUFFER_SIZE];
            while ((bytesRead = is.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            println "Downloaded file
to $localFileNameWithPathForStorage"
            return localFileNameWithPathForStorage
        } else {
            println "Could not find fileExtension header"
        }
    }
    }

    return ""
}

def listFiles() {
    println "***Listing files**"
    def restUrl=interopUrl+"/applicationsnapshots"
    def response

    response=openConnection(restUrl,"GET","")
    println "****"
}

def deleteFile(serverFileName) {
    println "***deleting file**"
    def restUrl=interopUrl+"/applicationsnapshots/"+serverFileName
    def response

    response=openConnection(restUrl,"DELETE","")
    println "****"
}

def getJobStatus(appName,jobId) {
    println "***get Job status**"
    def restUrl=planningUrl+"/applications/"+appName+"/jobs/" +
jobId

```

```

def response
response=openConnection(restUrl,"GET","")
println "****"
return response
}

def exportData(appName,jobName, exportServerFileName) {
println "***Exporting data**"
def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=EXPORT_DATA"
if (exportServerFileName!="") {
def exportFileJSON="{exportFileName:$exportServerFileName}"
restUrl=restUrl+"&parameters=" + exportFileJSON
}

def response
response=openConnection(restUrl,"POST","")
println "****"
return response
}

def getJobIdFromJSONResponse(response) {
def jobId=""
try {
Pattern regex = Pattern.compile("\"jobId\":\\d+");
Matcher matcher = regex.matcher(response);
while (matcher.find()) {
jobId = matcher.group(0).replace("\"jobId\":\"", "");
}
} catch (Exception e) {
println "No jobId found in server response"
}
return jobId
}

def downloadFile(serverFileName,localFolderForStorage) {
println "***Downloading file**"
def restUrl=interopUrl+"/applicationsnapshots/"+serverFileName+ "/"
contents"
def response

response=openConnection(restUrl,"GET",localFolderForStorage+"/"+serverFileNam
e)
println "****"
}

def exportMetaData(appName,jobName, exportServerFileName) {
println "***Exporting metadata**"
def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=EXPORT_METADATA"
if (exportServerFileName!="") {
def exportFileJSON="{exportZipFileName:$exportServerFileName}"
restUrl=restUrl+"&parameters=" + exportFileJSON
}
}

```

```

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def executeLCMExport(snapshotName) {
        println "***Exporting snapshot***"
        def typeExport="{type:export}"
        def restUrl=interopUrl+"/applicationsnapshots/"+snapshotName+
"/migration?q="+typeExport
        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def executeLCMImport(snapshotName) {
        println "***Importing snapshot***"
        def typeImport="{type:import}"
        def restUrl=interopUrl+"/applicationsnapshots/"+snapshotName+
"/migration?q="+typeImport
        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def runBusinessRule(appName,jobName, JSONRuntimePrompt) {
        println "***Running business rule***"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=RULES"
        if (JSONRuntimePrompt!="") {
            // Example for JSONRuntimePrompt {Period:Q1,Entity:USA}
            restUrl=restUrl+"&parameters=" + JSONRuntimePrompt
        }

        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def runRuleSet(appName,jobName) {
        println "***Running rule set***"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=RULESET"
        def response
        response=openConnection(restUrl,"POST","")
        println "****"
    }

    def cubeRefresh(appName,jobName) {
        println "***Refreshing cube***"
        def restUrl=planningUrl+"/applications/"+appName+"/jobs?
jobName=" + jobName + "&jobType=CUBE_REFRESH"
    }

```

```

def response
response=openConnection(restUrl,"POST","")
println "****"
}

def runPlanTypeMap(appName,jobName, clearData) {
println "***Running map (job of type plan_type_map)**"
def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=PLAN_TYPE_MAP"
if (clearData!=null && clearData.toUpperCase()=="FALSE") {
restUrl=restUrl+"&parameters={clearData:false}"
} else {
println "Clear data is set to true (default)"
}
def response
response=openConnection(restUrl,"POST","")
println "****"
}

def importData(appName,jobName, importFileName) {
println "***Importing data**"
def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=IMPORT_DATA"
if (importFileName!="") {
def exportFileJSON="{importFileName:$importFileName}"
restUrl=restUrl+"&parameters=" + exportFileJSON
}

def response
response=openConnection(restUrl,"POST","")
println "****"
}

def importMetaData(appName,jobName, importZipFileName) {
println "***Importing metadata**"
def restUrl=planningUrl+"/applications/"+appName+"/jobs?jobName=" +
jobName + "&jobType=IMPORT_METADATA"
if (importZipFileName!="") {
def exportFileJSON="{importZipFileName:$importZipFileName}"
restUrl=restUrl+"&parameters=" + exportFileJSON
}

def response
response=openConnection(restUrl,"POST","")
println "****"
}

def uploadFile(localPath,fileName) {
println "***Uploading file**"
def restUrl=interopUrl+"/applicationsnapshots/"+fileName

final int DEFAULT_CHUNK_SIZE = 50 * 1024 * 1024;
int packetNo = 1;
boolean status = true;
byte[] lastChunk = null;

```

```

        File f = new File(localPath+"/"+fileName);
        InputStream fis = null;
        long totalFileSize = f.length();
        boolean isLast = false;
        Boolean isFirst = true;
        boolean firstRetry = true;
        int lastPacketNo = (int) (Math.ceil(totalFileSize/ (double)
DEFAULT_CHUNK_SIZE));
        long totalbytesRead = 0;
        try {
            fis = new BufferedInputStream(new
FileInputStream(localPath+"/"+fileName));
            while (totalbytesRead < totalFileSize && status) {
                int nextChunkSize = (int) Math.min(DEFAULT_CHUNK_SIZE,
totalFileSize - totalbytesRead);
                if (lastChunk == null) {
                    lastChunk = new byte[nextChunkSize];
                    int bytesRead = fis.read(lastChunk);
                    totalbytesRead += bytesRead;
                    if (packetNo == lastPacketNo) {
                        isLast = true;
                    }
                    status =
sendRequestToRestForUpload(restUrl,isFirst, isLast,lastChunk);
                    isFirst=false;
                    if (status) {
                        println "\r" + ((100 * totalbytesRead)/
totalFileSize) + "% completed";
                    } else {
                        break;
                    }
                    packetNo = packetNo + 1;
                    lastChunk = null;
                }
            }
        } catch (Exception e) {
            println "Exception occurred while uploading file";
            println e.getMessage()
        } finally {
            if (null != fis) {
            }
        }
        println "*****"
    }

    def sendRequestToRestForUpload(restUrl,isFirst, isLast,lastChunk) {

        def url=restUrl+"/contents?
q={isLast:$isLast,chunkSize:"+lastChunk.length+",isFirst:$isLast}"
        println "Opening connection for upload to $url"
        int statusCode

        setProxyParams ()
        setSSLParams ()
    }

```



```

URL newUrl
newUrl=new URL(url)

connection = (URLConnection) newUrl.openConnection()

connection.setDoOutput(true)
connection.setDoInput(true)
connection.setUseCaches(false)
connection.setRequestMethod("POST")

connection.setRequestProperty("Content-Type","application/octet-
stream")

String userCredentials = domain + "." + user + ":" + pwd
String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)
connection.setRequestProperty("Authorization", basicAuth)
DataOutputStream wr = new
DataOutputStream(connection.getOutputStream());
wr.write(lastChunk);
wr.flush();

boolean status = false
int execStatus
try {
    execStatus = connection.getResponseCode();
    InputStream is = connection.getInputStream();
    BufferedReader br = new BufferedReader(new
InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line+"\n");
    }
    br.close();
    String stat = sb.toString();
    if (null == stat || stat.isEmpty()) {
        return status;
    } else {
        if (200 == execStatus) {
            println stat
        }
    }
} catch (Exception e) {
    println "Exception occurred while uploading file";
    println e.getMessage()
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}
}

```

```

// Helper functions

def unzip(fileName, destinationFolder) {
  // code from http://www.oracle.com/technetwork/articles/java/
  compress-1565076.html
  println ("**Unzipping "+fileName+"**")
  def fileList=[]
  int BUFFER = 2048;
  try {
    BufferedOutputStream dest = null;
    BufferedInputStream is = null;
    ZipEntry entry;
    ZipFile zipfile = new ZipFile(fileName);
    Enumeration e = zipfile.entries();
    while(e.hasMoreElements()) {
      entry = (ZipEntry) e.nextElement();
      //println("Extracting: " +entry);
      is = new
BufferedInputStream(zipfile.getInputStream(entry));
      int count;
      byte[] data;
      data = new byte[BUFFER];
      FileOutputStream fos = new
FileOutputStream(destinationFolder+"/"+entry.getName());
      fileList.push(entry.getName())
      dest = new BufferedOutputStream(fos, BUFFER);
      while ((count = is.read(data, 0, BUFFER)) != -1) {
        dest.write(data, 0, count);
      }
      dest.flush();
      dest.close();
      is.close();
    }
  } catch (FileNotFoundException fnfe) {
    println "Make sure there is not folder in the zip . Zip
not processed"
    //fnfe.printStackTrace();
  } catch(Exception e) {
    println "An error occurred while unzipping."
    println e.getMessage()

  }
  return fileList
  println "*****"
}

def findNbOfColsInCSV(filePath, fileName, delimiter) {
  File csvFile=new File (filePath+"/"+fileName);
  Scanner scanner = new Scanner(csvFile);
  scanner.useDelimiter(delimiter);

  def nbCols
  nbCols=0
  if (scanner.hasNextLine()) {

```

```

        String[] vals = scanner.nextLine().split(delimiter);
        nbCols=vals.size()
    }
    scanner.close();

    return nbCols
}

def getHeadersInCSVAsList(filePath, fileName, delimiter) {
    String[] headers =[]

    BufferedReader br = new BufferedReader(new
FileReader(filePath+"/"+fileName));
    String firstLine = br.readLine();
    println "First line is : " + firstLine
    println "Removing all non ascii chars from line"
    firstLine = firstLine.replaceAll("[^ -~]", "");
    firstLine = firstLine.replaceAll(" ", "");
    // firstLine = firstLine.replaceAll("\\", "");
    headers = firstLine.split(delimiter);

    def headersList = headers as List
    headersList = headersList.collect { it.trim() }

    return headersList
}

}

class WaitForCode {

static retry( sleepTime, nbOfRetries, Closure logicToRun){
    Throwable caught = null
    for(int i=0; i<nbOfRetries; i++){
        try {
            return logicToRun.call()
        } catch(Throwable t){
            caught = t
            println ("Retrying...")
            Thread.sleep(sleepTime)
        }
    }
    println ("Retry count limit exceeded. Stopping check.")
    throw caught
}
}
}

```

Groovy Sample – BicsRestClient.groovy

```

package com.oracle.ceal

import java.net.HttpURLConnection;
import javax.net.ssl.HostnameVerifier

```

```

import javax.net.ssl.HttpURLConnection
import javax.net.ssl.SSLContext
import javax.net.ssl.SSLSession
import javax.net.ssl.TrustManager
import javax.net.ssl.X509TrustManager

class BicsRestClient {
    private HttpURLConnection connection
    private bicsUrl
    private def proxyHost
    private def proxyPort
    private def user
    private def pwd
    private def domain
    private def ignoreSSLCertsErrors

    public BicsRestClient(bicsServerUrl,httpProxyHost, httpProxyPort,
identityDomain,username, password, ignoreSSLCertificationPathErrors) {
        bicsUrl=bicsServerUrl
        proxyHost=httpProxyHost
        proxyPort=httpProxyPort
        domain=identityDomain
        user=username
        pwd=password
        ignoreSSLCertsErrors=ignoreSSLCertificationPathErrors
    }

    def setProxyParams() {
        Properties systemProperties = System.getProperties()
        systemProperties.setProperty("http.proxyHost",proxyHost)
        systemProperties.setProperty("http.proxyPort",proxyPort)
        systemProperties.setProperty("https.proxyHost",proxyHost)
        systemProperties.setProperty("https.proxyPort",proxyPort)
    }

    def setSSLParams() {
        if (ignoreSSLCertsErrors !=null &&
ignoreSSLCertsErrors.toUpperCase()=="TRUE") {
            println "Ignoring SSL certification path errors"
            // Disable SSL cert validation

            def hostnameVerifier = [
                verify: { hostname, session -> true }
            ]
            def trustManager = [
                checkServerTrusted: { chain, authType -> },
                checkClientTrusted: { chain, authType -> },
                getAcceptedIssuers: { null }
            ]
        }
    }
}

```

```

HttpsURLConnection.setDefaultHostnameVerifier(hostnameVerifier as
HostnameVerifier)

HttpsURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory())

        SSLContext context = SSLContext.getInstance("SSL")
        context.init(null, [trustManager as X509TrustManager] as
TrustManager[], null)
    }
}

def openConnection(restUrl,method,contentype, body) {
println "Opening connection to bics $restUrl with method:$method"

    int statusCode

    setProxyParams()
    setSSLParams()

    URL newUrl
    newUrl=new URL(restUrl)

    connection = (HttpURLConnection) newUrl.openConnection()

    connection.setDoOutput(true)
    connection.setDoInput(true)
    connection.setUseCaches(false)
    if (method=="")
        connection.setRequestMethod("GET")
    else
        connection.setRequestMethod(method)

    //adding X-ID-TENANT-NAME <identity_domain>
    //connection.setRequestProperty("X-ID-TENANT-NAME",domain)

    if (contentype.toUpperCase()=="FORM") {
        connection.setRequestProperty("Content-Type","application/x-www-
form-urlencoded")
    }
    if (contentype.toUpperCase()=="JSON") {
        connection.setRequestProperty("Content-Type","application/json")
    }
    if (contentype.toUpperCase()=="") {
        // add no content type
    }

    String userCredentials = domain + "." + user + ":" + pwd
    String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)

    connection.setRequestProperty("Authorization", basicAuth)

    if (body!=null && body!="") {

```

```

        DataOutputStream wr = new DataOutputStream
(connection.getOutputStream ());
        wr.writeBytes (body);
        wr.flush ();
        wr.close ();
    }

    String response=""
    try {
        statusCode = connection.responseCode
        println "Connection status code: $statusCode "
        if (statusCode==401 || statusCode==403) {
            println "Not authorized"
        }
        if (statusCode==200) {
            println "Authentication succeeded"
            println "Server response:"
            println "-----"
            response=displayServerResponse(connection)
            println "-----"
        }
        if (statusCode==400 || statusCode==500) {
            println "Bad request"
            println "Server response:"
            println "-----"
            response=displayServerResponse(connection)
            println "-----"
        }
    } catch (Exception e) {
        println "Error connecting to the URL"
        println e.getMessage()
    } finally {
        if (connection != null) {
            connection.disconnect();
        }
    }

    return response
}

def displayServerResponse(connection) {
    InputStream is;
    if (connection.getResponseCode()==200) {
        is=connection.getInputStream();
    } else {
        is=connection.getErrorStream();
    }
    println "Response Content-Type:"+connection.getContentType()
    BufferedReader br = new BufferedReader(new
InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line+"\n");
    }
}

```

```

    }
    br.close();
    println sb
    return sb.toString()
}

def aboutBics() {
    println "***About bics**"
    def restUrl=bicsUrl+"/dataload/v1/about"

    def response
    response=openConnection(restUrl,"GET","FORM","")
    println "****"
}

def listAllTables() {
    //<URL>/dataload/v1/tables

    println "***List tables**"
    def restUrl=bicsUrl+"/dataload/v1/tables"

    def response
    response=openConnection(restUrl,"GET","FORM","")
    println "****"
}

def getTableInfo(tableName) {
    println "***Get table info**"
    def restUrl=bicsUrl+"/dataload/v1/tables?
name="+tableName.toUpperCase()

    def response
    response=openConnection(restUrl,"GET","","")
    println "****"
}

def createTableToLoadCSV(tableName, numberOfVarCharCols, columnPrefix ) {
    println "***Create table**"
    // create json manually for X columns

    /*
    {
    "columnName":"COL_1",
    "dataType":"VARCHAR",
    "length":300,
    "precision":0,
    "nullable":true,
    "defaultValue":null,
    },
    * */

    def restUrl=bicsUrl+"/dataload/v1/tables/"+tableName.toUpperCase()

```

```

def JSONColumns

JSONColumns="["
def i
for (i = 1; i <=numberOfVarCharCols; i++) {
    if (i==numberOfVarCharCols) {

JSONColumns=JSONColumns+"\columnName\":"+""+columnPrefix.toUpperCase()
+""+i+"\", \"dataType\":"\"VARCHAR\", \"length\":300, \"precision\":0, \"null
able\":true, \"defaultValue\":null}"
    } else {

JSONColumns=JSONColumns+"\columnName\":"+""+columnPrefix.toUpperCase()
+""+i+"\", \"dataType\":"\"VARCHAR\", \"length\":300, \"precision\":0, \"null
able\":true, \"defaultValue\":null}, "
    }
}
JSONColumns=JSONColumns+"]"

println "JSON columns:"+JSONColumns
def response
response=openConnection(restUrl,"PUT","JSON",JSONColumns)
println "*****"
}

def createTableToLoadCSVWithHeaderNames(tableName, listHeaders ) {
println "***Create table***"
// create json manually for X columns with headers in list

/*
{
"columnName":"COL_1",
"dataType":"VARCHAR",
"length":300,
"precision":0,
"nullable":true,
"defaultValue":null,
},
* */

def restUrl=bicsUrl+"/dataload/v1/
tables/"+tableName.toUpperCase()

def JSONColumns

JSONColumns="["

listHeaders.each { headerName ->
    if(headerName == listHeaders.last()) {

JSONColumns=JSONColumns+"\columnName\":"+""+headerName.toUpperCase()
+""+\", \"dataType\":"\"VARCHAR\", \"length\":300, \"precision\":0, \"nullable
\":true, \"defaultValue\":null}"

```



```

        } else {

JSONColumns=JSONColumns+"{\"columnName\": \""+headerName.toUpperCase()
+"\", \"dataType\": \"VARCHAR\", \"length\": 300, \"precision\": 0, \"nullable\": tru
e, \"defaultValue\": null}, "
        }
    }

JSONColumns=JSONColumns+"]"

println "JSON columns:"+JSONColumns
def response
response=openConnection(restUrl,"PUT","JSON",JSONColumns)
println "****"

}

def loadDataInTableUsingCSV(tableName, localCsvFilePath,
localCsvFileName,
delimiterInCsv,numberOfColumnsInCsv,numberOfLinesToSkip,columnPrefixInTable,i
sZipped) {
    println "***Load csv file in table**"
    println "Processing:"+localCsvFilePath+"/"+localCsvFileName

    if (isZipped==true) println "Upload of zip not supported at this
time. Ignoring isZipped parameter"

    File localCsv=new File(localCsvFilePath+"/"+localCsvFileName)
    if(!localCsv.exists() || localCsv.isDirectory()) {
        println "File does not exist"
        println "****"
        return
    }

def restUrl=bicsUrl+"/dataload/v1/tables/"+tableName.toUpperCase()+"/
data"

setProxyParams()
setSSLParams()

URL newUrl
newUrl=new URL(restUrl)

connection = (URLConnection) newUrl.openConnection()

connection.setDoOutput(true)
connection.setDoInput(true)
connection.setUseCaches(false)
connection.setRequestMethod("PUT")
//connection.setRequestProperty("X-ID-TENANT-NAME",domain)
String userCredentials = domain + "." + user + ":" + pwd
String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)

    connection.setRequestProperty("Authorization", basicAuth)

```

```

/*
 * The first part is a JSON descriptor (Content-Type:
application/json)
 * of the data load. The second part is an input stream
 * (Content-Type: application/octet-stream).
 * Data in the stream can be text data read
 * from comma-separated values (CSV)
 * */

def boundary = System.currentTimeMillis() ;
connection.setRequestProperty("Content-Type", "multipart/mixed;
boundary=" + boundary);
OutputStream outputStream = connection.getOutputStream();
PrintWriter writer = new PrintWriter(new
OutputStreamWriter(outputStream, "UTF-8"), true);

// JSON
/*
 * {
"columnMaps":[
    {
        "column":{
            "name":"NAME",
            "optionalJavaSqlType":null,
            "partOfUniqueKey":true,
        },
        "position":1,
    },
    {...
    }
],
"optionalMaximumErrors":null,
"removeDuplicates":true
"optionalWriteMode":"Insert all",
"delimiter":",",
"timestampFormat":"yyyy-MM-dd",
"numberOfLinesToSkip":0
},
*
*
*/
def i
def JSONDataLoad
JSONDataLoad="{\"columnMaps\":["
for (i =1; i <=numberOfColumnsInCsv; i++) {
    if (i==numberOfColumnsInCsv) {
        JSONDataLoad=JSONDataLoad+{\"column\":
{\"name\":\")+columnPrefixInTable.toUpperCase()
+\""+i+"\", "+\"optionalJavaSqlType\":null, \"partOfUniqueKey\":false}, "+
\"position\":"+i+"}
    } else {
        JSONDataLoad=JSONDataLoad+{\"column\":
{\"name\":\")+columnPrefixInTable.toUpperCase()
+\""+i+"\", "+\"optionalJavaSqlType\":null, \"partOfUniqueKey\":false}, "+

```

```

"\position\":"+i+"},"
    }
    }
    JSONDataLoad=JSONDataLoad+'''],
    "optionalMaximumErrors":null,
    "removeDuplicates":false,
    "optionalWriteMode":"Insert all",
    "delimiter":'''+delimiterInCsv+"\","+''
    "timestampFormat":"","
    "numberOfLinesToSkip":'' + numberOfLinesToSkip +''}
    '''

    writer.append("--" + boundary).append("\r\n");
    writer.append("Content-Type: application/json").append("\r\n");
    writer.append("\r\n");
    writer.flush();
    writer.append(JSONDataLoad)
    writer.append("\r\n");
    writer.flush();
    writer.append("\r\n").flush();
    //writer.append("--" + boundary ).append("\r\n");

    // CSV or ZIP file content
    writer.append("--" + boundary).append("\r\n");
    writer.append("Content-Type: application/octet-
stream").append("\r\n");
    writer.append("\r\n");
    writer.flush();

    FileInputStream inputStream = new FileInputStream(new
File(localCsvFilePath+"/"+localCsvFileName));
    byte[] buffer = new byte[4096];
    int bytesRead = -1;
    while ((bytesRead = inputStream.read(buffer)) != -1) {
        outputStream.write(buffer, 0, bytesRead);
    }
    outputStream.flush();
    inputStream.close();

    writer.append("\r\n");
    writer.flush();

    writer.append("\r\n").flush();
    writer.append("--" + boundary + "--").append("\r\n");
    writer.close();

    String response=""
    def statusCode
    try {
        statusCode = connection.responseCode
        println "Connection status code: $statusCode "
        if (statusCode==401 || statusCode==403) {
            println "Not authorized"
        }
        if (statusCode==200) {

```

```

        println "Authentication succeeded"
        println "Server response:"
        println "-----"
        response=displayServerResponse(connection)
        println "-----"
    }
    if (statusCode==400 || statusCode==500) {
        println "Bad request"
        println "Server response:"
        println "-----"
        response=displayServerResponse(connection)
        println "-----"
    }
} catch (Exception e) {
    println "Error connecting to the URL"
    println e.getMessage()
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}

println "****"
}

def loadDataInTableUsingCSVAndHeader(tableName, localCsvFilePath,
localCsvFileName,
delimiterInCsv,numberOfLinesToSkip,listHeaders,isZipped) {
    println "***Load csv file in table using headers***"
    println "Processing:"+localCsvFilePath+"/"+localCsvFileName

    if (isZipped==true) println "Upload of zip not supported at
this time. Ignoring isZipped parameter"

    File localCsv=new File(localCsvFilePath+"/"+localCsvFileName)
    if(!localCsv.exists() || localCsv.isDirectory()) {
        println "File does not exist"
        println "****"
        return
    }

    def restUrl=bicsUrl+"/dataload/v1/
tables/"+tableName.toUpperCase()+"/data"

    setProxyParams()
    setSSLParams()

    URL newUrl
    newUrl=new URL(restUrl)

    connection = (URLConnection) newUrl.openConnection()

    connection.setDoOutput(true)
    connection.setDoInput(true)
    connection.setUseCaches(false)

```

```

connection.setRequestMethod("PUT")
//connection.setRequestProperty("X-ID-TENANT-NAME",domain)
String userCredentials = domain + "." + user + ":" + pwd
String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes()
)
connection.setRequestProperty("Authorization", basicAuth)

/*
 * The first part is a JSON descriptor (Content-Type: application/
json)
 * of the data load. The second part is an input stream
 * (Content-Type: application/octet-stream).
 * Data in the stream can be text data read
 * from comma-separated values (CSV)
 * */

def boundary = System.currentTimeMillis() ;
connection.setRequestProperty("Content-Type", "multipart/mixed;
boundary=" + boundary);
OutputStream outputStream = connection.getOutputStream();
PrintWriter writer = new PrintWriter(new
OutputStreamWriter(outputStream, "UTF-8"), true);

// JSON
/*
 * {
"columnMaps":[
    {
        "column":{
            "name":"NAME",
            "optionalJavaSqlType":null,
            "partOfUniqueKey":true,
        },
        "position":1,
    },
    {...
    }
],
"optionalMaximumErrors":null,
"removeDuplicates":true
"optionalWriteMode":"Insert all",
"delimiter":",",
"timestampFormat":"yyyy-MM-dd",
"numberOfLinesToSkip":0
},
 *
 *
 */

int i
i=1
def JSONDataLoad
JSONDataLoad="{\"columnMaps\":["

```

```

        listHeaders.each { headerName ->
            if(headerName == listHeaders.last()) {
                JSONDataLoad=JSONDataLoad+"{\"column\":
{ \"name\": \""+headerName.toUpperCase()
+"\", \"optionalJavaSqlType\": null, \"partOfUniqueKey\": false}, \"position\": "+i+"}"
            } else {
                JSONDataLoad=JSONDataLoad+"{\"column\":
{ \"name\": \""+headerName.toUpperCase()
+"\", \"optionalJavaSqlType\": null, \"partOfUniqueKey\": false}, \"position\": "+i+"}, "
            }
            i=i+1
        }

JSONDataLoad=JSONDataLoad+''',
"optionalMaximumErrors": null,
"removeDuplicates": false,
"optionalWriteMode": "Insert all",
"delimiter": ""'+delimiterInCsv+"\", \"+'''
"timestampFormat": "",
"numberOfLinesToSkip": '' + numberOfLinesToSkip + '''
'''

writer.append("--" + boundary).append("\r\n");
writer.append("Content-Type: application/json").append("\r\n");
writer.append("\r\n");
writer.flush();
writer.append(JSONDataLoad)
writer.append("\r\n");
writer.flush();
writer.append("\r\n").flush();
//writer.append("--" + boundary ).append("\r\n");

// CSV or ZIP file content
writer.append("--" + boundary).append("\r\n");
writer.append("Content-Type: application/octet-
stream").append("\r\n");
writer.append("\r\n");
writer.flush();

FileInputStream inputStream = new FileInputStream(new
File(localCsvFilePath+"/"+localCsvFileName));
byte[] buffer = new byte[4096];
int bytesRead = -1;
while ((bytesRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, bytesRead);
}
outputStream.flush();
inputStream.close();

writer.append("\r\n");
writer.flush();

```

```

writer.append("\r\n").flush();
writer.append("--" + boundary + "--").append("\r\n");
writer.close();

String response=""
def statusCode
try {
    statusCode = connection.responseCode
    println "Connection status code: $statusCode "
    if (statusCode==401 || statusCode==403) {
        println "Not authorized"
    }
    if (statusCode==200) {
        println "Authentication succeeded"
        println "Server response:"
        println "-----"
        response=displayServerResponse(connection)
        println "-----"
    }
    if (statusCode==400 || statusCode==500) {
        println "Bad request"
        println "Server response:"
        println "-----"
        response=displayServerResponse(connection)
        println "-----"
    }
} catch (Exception e) {
    println "Error connecting to the URL"
    println e.getMessage()
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}

println "****"
}

def deleteTable(tableName) {
    println "***Delete table**"
    def restUrl=bicsUrl+"/dataload/v1/tables/"+tableName.toUpperCase()

    def response
    response=openConnection(restUrl,"DELETE","","")
    println "****"
}

def deleteDataFromTable(tableName) {
    println "***Delete all data from table**"
    def restUrl=bicsUrl+"/dataload/v1/tables/"+tableName.toUpperCase()+"/
data"

    def response
    response=openConnection(restUrl,"DELETE","","")

```

```

        println "****"
    }

    def truncateList(listName, truncateLength) {
        println "***Truncating list**"
        def trimmedList
        listName=listName*.trim()
        trimmedList=listName*.take(truncateLength)

        println ("New list:"+trimmedList)
        println "****"
        return trimmedList
    }
}

```

Groovy Sample – ApexRestClient.groovy

```

package com.oracle.ceal

import java.net.HttpURLConnection;
import javax.net.ssl.HostnameVerifier
import javax.net.ssl.HttpURLConnection
import javax.net.ssl.SSLContext
import javax.net.ssl.SSLSession
import javax.net.ssl.TrustManager
import javax.net.ssl.X509TrustManager

class ApexRestClient {

    private HttpURLConnection connection
    private apexUrl
    private def proxyHost
    private def proxyPort
    private def user
    private def pwd
    private def domain
    private def ignoreSSLCertsErrors

    public ApexRestClient(apexServerUrl,httpProxyHost, httpProxyPort,
identityDomain,username, password, ignoreSSLCertificationPathErrors) {
        apexUrl=apexServerUrl
        proxyHost=httpProxyHost
        proxyPort=httpProxyPort
        domain=identityDomain
        user=username
        pwd=password
        ignoreSSLCertsErrors=ignoreSSLCertificationPathErrors
    }

    def setProxyParams() {
        Properties systemProperties = System.getProperties()
        systemProperties.setProperty("http.proxyHost",proxyHost)
        systemProperties.setProperty("http.proxyPort",proxyPort)
    }
}

```



```

        systemProperties.setProperty("https.proxyHost",proxyHost)
        systemProperties.setProperty("https.proxyPort",proxyPort)
    }

    def setSSLParams() {
        if (ignoreSSLCertsErrors !=null &&
ignoreSSLCertsErrors.toUpperCase()=="TRUE") {
            println "Ignoring SSL certification path errors"
            // Disable SSL cert validation

            def hostnameVerifier = [
                verify: { hostname, session -> true }
            ]
            def trustManager = [
                checkServerTrusted: { chain, authType -> },
                checkClientTrusted: { chain, authType -> },
                getAcceptedIssuers: { null }
            ]

            HttpsURLConnection.setDefaultHostnameVerifier(hostnameVerifier
as HostnameVerifier)

            HttpsURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory())

            SSLContext context = SSLContext.getInstance("SSL")
            context.init(null, [trustManager as X509TrustManager] as
TrustManager[], null)
        }
    }

    def openConnection(restUrl,method,contentype, body) {
        println "Opening connection to apex $restUrl with method:$method"

        int statusCode

        setProxyParams()
        setSSLParams()

        URL newUrl
        newUrl=new URL(restUrl)

        connection = (HttpURLConnection) newUrl.openConnection()

        connection.setDoOutput(true)
        connection.setDoInput(true)
        connection.setUseCaches(false)
        if (method=="")
            connection.setRequestMethod("GET")
        else
            connection.setRequestMethod(method)
    }

```

```

        if (contentType.toUpperCase()=="FORM") {
            connection.setRequestProperty("Content-
Type","application/x-www-form-urlencoded")
        }
        if (contentType.toUpperCase()=="JSON") {
            connection.setRequestProperty("Content-Type","application/
json")
        }
        if (contentType.toUpperCase()=="") {
            // add no content type
        }

        String userCredentials = domain + "." + user + ":" + pwd
        String basicAuth = "Basic " +
javax.xml.bind.DatatypeConverter.printBase64Binary(userCredentials.getBytes())
        connection.setRequestProperty("Authorization", basicAuth)

        if (body!=null && body!="") {
            DataOutputStream wr = new DataOutputStream
(connection.getOutputStream ());
            wr.writeBytes (body);
            wr.flush ();
            wr.close ();
        }

        String response=""
        try {
            statusCode = connection.responseCode
            println "Connection status code: $statusCode "
            if (statusCode==401 || statusCode==403) {
                println "Not authorized"
            }
            if (statusCode==200) {
                println "Authentication succeeded"
                println "Server response:"
                println "-----"
                response=displayServerResponse(connection)
                println "-----"
            }
            if (statusCode==400 || statusCode==500) {
                println "Bad request"
                println "Server response:"
                println "-----"
                response=displayServerResponse(connection)
                println "-----"
            }
        } catch (Exception e) {
            println "Error connecting to the URL"
            println e.getMessage()
        } finally {
            if (connection != null) {

```

```

        connection.disconnect();
    }
}

return response
}

def displayServerResponse(connection) {
    InputStream is;
    if (connection.getResponseCode()==200) {
        is=connection.getInputStream();
    } else {
        is=connection.getErrorStream();
    }
    println "Response Content-Type:"+connection.getContentType()
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line+"\n");
    }
    br.close();
    println sb
    return sb.toString()
}

def launchProcUsingGET(apexUri) {
    println "***Launching PL/SQL in apex**"

    def restUrl=apexUrl+"/"+apexUri

    /*
    * Procedure in apex is defined this way
    * RESTful Service Module: bics/
    * URI Template: plsqli/
    * Method: GET
    * Source Type: PL/SQL
    * Requires Secure Access: YES
    * Source:

        DECLARE
            prevdeptno    number;
            deptloc       varchar2(30);
            deptname      varchar2(30);
            CURSOR getemps IS select * from emp
                               where ((select job from emp where
ename = :empname) IN ('PRESIDENT', 'MANAGER'))
                               or deptno = (select deptno from emp
where ename = :empname)
                               order by deptno, ename;

        BEGIN
            sys.htp.htmllopen;
            sys.htp.headopen;
            sys.htp.title('Departments');
            sys.htp.headclose;

```

```

        sys.htp.bodyopen;

        for empregs in getemps
        loop
            if empregs.deptno != prevdeptno or prevdeptno
is null then
                select dname, loc into deptname, deptloc
                from dept where deptno = (select deptno
from emp where ename = empregs.ename);
                if prevdeptno is not null then
                    sys.htp.print('</ul>');
                end if;
                sys.htp.print('Department ' || deptname || '
located in ' || deptloc || '<p/>');
                sys.htp.print('<ul>');
            end if;
            sys.htp.print('<li>');

            prevdeptno := empregs.deptno;
        end loop;
        sys.htp.print('</ul>');
        sys.htp.bodyclose;
        sys.htp.htmlclose;
    END;

```

URL call will be in the form: https://
<SERVER>.oraclecloudapps.com/apex/bics/plsql/
Response will be in following format for this specific
plsql example

```

Response Content-Type:text/html; charset=UTF-8
<HTML>
<HEAD>
<TITLE>Departments</TITLE>
</HEAD>
<BODY>
</ul>
</BODY>
</HTML>

*/
def response
response=openConnection(restUrl,"GET","FORM","")
println "****"

println "****"
}
def launchSQLQueryUsingGETAndVariableOnUrl(apexUri, parameter) {
println "**Launching Sql query in apex**"

/*
* Procedure in apex is defined this way
* RESTful Service Module: bics/
  URI Template: test/{ID}
  Method: GET
  Source Type: Query

```

```

        Format: JSON
        Requires Secure Access: YES
        Source: select
EMPNO,ENAME, JOB,MGR, HIREDATE, SAL, COMM, DEPTNO
        from EMP where EMPNO = :ID

        URL call will be in the form: https://
<SERVER>.oraclecloudapps.com/apex/bics/test/7839
        Response will be in following format:
        Response Content-Type:application/json
        {"next":{"$ref":"https://<SERVER>.oraclecloudapps.com/apex/
bics/test/7839?page=1"},"items":
[{"empno":7839,"ename":"KING","job":"PRESIDENT","hiredate":"1981-11-17T00:00:
00Z","sal":5000,"deptno":10}]}

        *
        */

def restUrl=apexUrl+"/"+apexUri+"/"+parameter

def response
response=openConnection(restUrl,"GET","FORM","")
println "*****"
    }
}

```

Groovy Sample — APEXAutomationParameters.groovy

```

package com.oracle.ceal

import java.io.File;
import java.util.Properties;

class APEXAutomationParameters {
    private Properties props = new Properties()

    def apexRestUrl
    def APEX_REST_URL='apexRestUrl'
    def identityDomain
    def APEX_IDENTITY_DOMAIN='apexIdentityDomain'
    def username
    def APEX_USERNAME='apexUsername'
    def password
    def APEX_PASSWORD='apexPassword'
    def proxyHost
    def PROXY_HOST='proxyHost'
    def proxyPort
    def PROXY_PORT='proxyPort'
    def ignoreSSLCertificationPathErrors
    def IGNORE_CERT_PATH_ERRORS='ignoreSSLCertificationPathErrors'

    public APEXAutomationParameters(propertiesFile) {
        def propsFileName=propertiesFile
        File propsFile = new File(propsFileName)
    }
}

```

```

    try {
        props.load(propsFile.newDataInputStream())
    } catch ( FileNotFoundException fnfe) {
        println "$propsFileName APEX properties file not found in
the current
directory. Exiting."
        System.exit(1);
    }
    apexRestUrl=props.getProperty(APEX_REST_URL)
    identityDomain=props.getProperty(APEX_IDENTITY_DOMAIN)
    username=props.getProperty(APEX_USERNAME)
    password=props.getProperty(APEX_PASSWORD)
    proxyHost=props.getProperty(PROXY_HOST)
    proxyPort=props.getProperty(PROXY_PORT)

ignoreSSLCertificationPathErrors=props.getProperty(IGNORE_CERT_PATH_ERR
ORS)

}

def isConfigValid() {
    try {
        // Required parameters check
        assert apexRestUrl != '' : "$APEX_REST_URL is empty"
        assert identityDomain != '' : "$APEX_IDENTITY_DOMAIN is
empty"

        assert username != '' : "$APEX_USERNAME is empty"
        assert password != '' : "$APEX_PASSWORD is empty"

        // validate url is correct
        URL apexUrl=new URL(apexRestUrl)
        // connection test

        // ssl check

        println "$APEX_REST_URL = $apexRestUrl"
        println "$APEX_IDENTITY_DOMAIN = $identityDomain"
        println "$APEX_USERNAME = $username"
        println "$APEX_PASSWORD = *****"

    } catch(AssertionError e) {
        println e.getMessage()
        return false
    } catch (MalformedURLException e) {
        println "APEX Rest url is incorrect. Current
value:$apexRestUrl , expected format http|https://"
        println e.getMessage()
        return false
    }
    return true
}
}

```

Groovy Sample — BICSAutomationParameters.groovy

```

package com.oracle.ceal

import java.io.File;
import java.util.Properties;

class BICSAutomationParameters {

    private Properties props = new Properties()

    def bicsRestUrl
    def BICS_REST_URL='bicsRestUrl'
    def identityDomain
    def BICS_IDENTITY_DOMAIN='bicsIdentityDomain'
    def username
    def BICS_USERNAME='bicsUsername'
    def password
    def BICS_PASSWORD='bicsPassword'
    def proxyHost
    def PROXY_HOST='proxyHost'
    def proxyPort
    def PROXY_PORT='proxyPort'
    def ignoreSSLCertificationPathErrors
    def IGNORE_CERT_PATH_ERRORS='ignoreSSLCertificationPathErrors'

    public BICSAutomationParameters(propertiesFile) {
        def propsFileName=propertiesFile
        File propsFile = new File(propsFileName)
        try {
            props.load(propsFile.newDataInputStream())
        } catch ( FileNotFoundException fnfe) {
            println "$propsFileName BICS properties file not found in the
current directory. Exiting."
            System.exit(1);
        }
        bicsRestUrl=props.getProperty(BICS_REST_URL)
        identityDomain=props.getProperty(BICS_IDENTITY_DOMAIN)
        username=props.getProperty(BICS_USERNAME)
        password=props.getProperty(BICS_PASSWORD)
        proxyHost=props.getProperty(PROXY_HOST)
        proxyPort=props.getProperty(PROXY_PORT)

        ignoreSSLCertificationPathErrors=props.getProperty(IGNORE_CERT_PATH_ERRORS)
    }

    def isConfigValid() {
        try {
            // Required parameters check
            assert bicsRestUrl != '' : "$BICS_REST_URL is empty"
            assert identityDomain != '' : "$BICS_IDENTITY_DOMAIN is empty"
            assert username != '' : "$BICS_USERNAME is empty"
            assert password != '' : "$BICS_PASSWORD is empty"
        }
    }
}

```

```

        // validate url is correct
        URL bicsUrl=new URL(bicsRestUrl)
        // connection test

        // ssl check

        println "$BICS_REST_URL = $bicsRestUrl"
        println "$BICS_IDENTITY_DOMAIN = $identityDomain"
        println "$BICS_USERNAME = $username"
        println "$BICS_PASSWORD = *****"

    } catch(AssertionError e) {
        println e.getMessage()
        return false
    } catch (MalformedURLException e) {
        println "BICS Rest url is incorrect. Current
value:$bicsRestUrl , expected format http|https://"
        println e.getMessage()
        return false
    }
    return true
}
}
}

```

Groovy Sample — PBCSAutomationParameters.groovy

```

package com.oracle.ceal

class PBCSAutomationParameters {
    private Properties props = new Properties()

    def planningRestUrl
    def PBCS_PLANNING_REST_URL='pbcsPlanningRestUrl'
    def interopRestUrl
    def PBCS_INTEROP_REST_URL='pbcsInteropRestUrl'
    def identityDomain
    def PBCS_IDENTITY_DOMAIN='pbcsIdentityDomain'
    def username
    def PBCS_USERNAME='pbcsUsername'
    def password
    def PBCS_PASSWORD='pbcsPassword'
    def proxyHost
    def PROXY_HOST='proxyHost'
    def proxyPort
    def PROXY_PORT='proxyPort'
    def ignoreSSLCertificationPathErrors
    def IGNORE_CERT_PATH_ERRORS='ignoreSSLCertificationPathErrors'

    public PBCSAutomationParameters(propertiesFile) {

```



```

def propsFileName=propertiesFile
File propsFile = new File(propsFileName)
try {
    props.load(propsFile.newDataInputStream())
} catch ( FileNotFoundException fnfe) {
    println "$propsFileName PBCS properties file not found in the
current directory. Exiting."
    System.exit(1);
}
planningRestUrl=props.getProperty(PBCS_PLANNING_REST_URL)
interopRestUrl=props.getProperty(PBCS_INTEROP_REST_URL)
identityDomain=props.getProperty(PBCS_IDENTITY_DOMAIN)
username=props.getProperty(PBCS_USERNAME)
password=props.getProperty(PBCS_PASSWORD)
proxyHost=props.getProperty(PROXY_HOST)
proxyPort=props.getProperty(PROXY_PORT)

ignoreSSLCertificationPathErrors=props.getProperty(IGNORE_CERT_PATH_ERRORS)

}

def isConfigValid() {
    try {
        // Required parameters check
        assert planningRestUrl != '' : "$PBCS_PLANNING_REST_URL is empty"
        assert interopRestUrl != '' : "$PBCS_INTEROP_REST_URL is empty"
        assert identityDomain != '' : "$PBCS_IDENTITY_DOMAIN is empty"
        assert username != '' : "$PBCS_USERNAME is empty"
        assert password != '' : "$PBCS_PASSWORD is empty"

        // validate url is correct
        URL planningUrl=new URL(planningRestUrl)
        URL interopUrl=new URL(interopRestUrl)
        // connection test

        // ssl check

        println "$PBCS_PLANNING_REST_URL = $planningRestUrl"
        println "$PBCS_INTEROP_REST_URL = $interopRestUrl"
        println "$PBCS_IDENTITY_DOMAIN = $identityDomain"
        println "$PBCS_USERNAME = $username"
        println "$PBCS_PASSWORD = *****"

    } catch(AssertionError e) {
        println e.getMessage()
        return false
    } catch (MalformedURLException e) {
        println "PBCS Rest urls are incorrect. Current
value:$planningRestUrl and $interopRestUrl, expected format http|https://
<SERVER>/HyperionPlanning/rest/11.1.2.3.xyz http|https://<SERVER>/interop/
rest/11.1.2.3.xyz"
        println e.getMessage()
        return false
    }
}

```

```
        }  
        return true  
    }  
  
}
```

Troubleshooting the Integration

Use an HTTP proxy such as Fiddler to trace HTTP calls

- In this case, set `proxyHost / proxyPort` to `localhost 8888` in your properties file defining the configuration
- Also set `ignoreSSLCertificationPathErrors=true`