Oracle® Fusion Cloud EPM

Working with EPM Automate for Oracle Enterprise Performance Management Cloud

E96247-78

Oracle Fusion Cloud EPM Working with EPM Automate for Oracle Enterprise Performance Management Cloud,

E96247-78

Copyright © 2016, 2024, 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, MySQL, and NetSuite 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

3 Command Execution Sample Scenarios

A Preparing to Run the simulateConcurrentUsage Command

B Preparing to Run the Replay Command

C Handling Special Characters

D Commands Specific to Each EPM Cloud Service

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

About EPM Automate

EPM Automate enables users to remotely perform tasks within Oracle Enterprise Performance Management Cloud environments.

EPM Cloud Service Administrators can automate many repeatable tasks including the following:

- Import and export metadata, data, artifact and application snapshots, templates, and Data Management mappings
- Upload files into environments, list files, and delete files from the service
- Download snapshots, reports, and metadata and data files from the service
- Run business rules on data, and refresh the application
- Copy data from one database to another; typically, from a block storage database to an aggregate storage database or from a block storage database to another block storage database
- Run a Data Management batch rule
- Generate Data Management reports, provisioning report, and user audit report
- Import pre-mapped balance data, currency rates, pre-mapped transactions, balances data, and profiles into Account Reconciliation
- Copy profiles to a period to initiate the reconciliation process
- Deploy the calculation cube of Profitability and Cost Management applications
- Clear, copy, and delete Point of Views in Enterprise Profitability and Cost Management and Profitability and Cost Management applications
- Replay Oracle Smart View for Office or REST API load on an environment to enable performance testing under heavy load
- Import supplemental data from a file into Financial Consolidation and Close

You can create scripts that are capable of completing a wide array of tasks and automate their execution using a scheduler. For example, you can create a script to download the daily maintenance backup from environments to create local backups of your artifacts and data.

 Tutorial: How to execute Planning commands using EPM Automate

Installing EPM Automate

You install EPM Automate to run commands. Some commands can also be run directly in Oracle Enterprise Performance Management Cloud using Groovy scripts without installing EPM Automate.

EPM Automate installer for Windows and Linux/UNIX, and macOS X is available from your EPM Cloud environment.

Because Windows versions 10 and newer permits only Windows administrators to install EPM Automate, it can be installed and upgraded only by Windows administrators. EPM Automate can be upgraded by the user who installed it or by another Windows administrator.

In this section:

Capacity and Port Requirements

Because EPM Automate is a light-weight client, it does not require a large client footprint. All processing takes place in Oracle Enterprise Performance Management Cloud.

You can install EPM Automate on standard client machines, virtual machines, and Oracle Integration Cloud machines that can access external hosts over a secure HTTP connection.

EPM Automate connects to EPM Cloud using the standard TLS port (port 443). You do not need to open additional outgoing ports for EPM Automate.

At this time, EPM Automate does not support mutual TLS (mTLS) authentication.

Supported Platforms

EPM Automate can be installed on virtual machines and Oracle Integration Cloud (OIC) machines that can access external hosts over a secure HTTP connection.

Note:

- EPM Automate may be used only on 64-bit operating systems that are currently supported by the operating system vendor.
- EPM Automate does not work with SOCKS proxy; it works only with HTTP/HTTPS proxy.
- EPM Automate supports Basic, Digest, Kerberos, Negotiate, and NTLM authentication mechanisms to connect to the proxy server.
- EPM Automate can connect to Oracle Enterprise Performance Management Cloud through API Gateways, such as Google APIGEE, IBM Data Power, and other reverse proxy servers.
  For this to work, configure the gateway or reverse proxy by setting the target as the URL of your EPM Cloud environment without any context such as `/epmcloud`. Example: `https://epm-idDomain.epm.dataCenterRegion.oraclecloud.com`. Then, use the reverse proxy URL instead of the EPM Cloud URL in the login command. For configuration information, see the documentation of your gateway or proxy server.

  While configuring the proxy settings, be sure to pass the response code from EPM Cloud to EPM Automate without modifying it in any manner to allow EPM Automate to correctly process response codes such as 200,206, 400, 404, 500, 501, and so on. For example, for IBM Datapower, set `proxy HTTP Response` to `ON`. Additionally, the API gateway should allow HTTP methods (GET, POST, PUT, PATCH, and DELETE).

On Linux and UNIX computers, EPM Automate looks for the following environment variables to determine HTTP or HTTPS proxy settings:

- proxyHost
- proxyPort

Examples of http proxy settings:

```
export proxyHost=host.example.com

export proxyPort=8000
```

Examples of https proxy settings:

```
export proxyHost=host.example.com

export proxyPort=8080
```

> **Note:**
>
> EPM Automate can use the OAuth 2.0 authentication protocol to access OCI (Gen 2) EPM Cloud environments (if configured for OAuth) to execute commands, especially for automating the running of commands.
>
> In Classic environments and those OCI (Gen 2) environments that use basic authentication, EPM Automate does not work with corporate SSO (identity provider) credentials. Because users cannot sign in using corporate credentials, the user accounts for accessing EPM Automate must be maintained in the identity domain. If your subscription is configured for SSO, you must also enable EPM Automate users to sign-in with their identity domain credentials. See Enabling Sign In With Identity Domain Credentials in *Administering Oracle Cloud Identity Management*.

Download Instructions: Downloading and Installing Clients in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators.*

Java Runtime Environment and EPM Automate

Installing EPM Automate on Windows installs the required Java Runtime Environment (JRE). However, a JRE is not included in the Linux, Unix, and macOS X installers. You must have access to a JRE installation (version 8 through version 11) to use EPM Automate.

You are entitled to use Oracle Java Standard Edition (SE) with EPM Automate without the need to separately purchase a Java SE subscription. For details about Oracle JDK licensing with EPM Automate, refer to Oracle Support Document 1557737.1: "Support Entitlement for Java SE When Used As Part of Another Oracle Product" .

Using OpenJDK

You may use OpenJDK version 14 or higher instead of JRE on Linux, Unix, and macOS X platforms.

OpenJDK, Oracle's free, GPL-licensed, production-ready JDK, can be downloaded from https://openjdk.java.net. Instructions to install OpenJDK are also available on this web site.

Before starting an EPM Automate session, set the `JAVA_HOME` environment variable to point to your OpenJDK installation:

macOS X example (Bash shell assumed) to use OpenJDK version 14 installed in your home directory.

```
cd ~/
export JAVA_HOME=$(/usr/jdk-14.jdk/Contents/Home)
```

Linux example (Bash shell assumed) to use OpenJDK version 14 installed in your home directory

```
cd ~/
export JAVA_HOME=/openjdk/jdk-14.0.2
```

Windows Instructions

By default, EPM Automate is installed in `C:/Oracle/EPM Automate`.

To install EPM Automate:

1. From the Windows computer where you want to install EPM Automate, access an environment.
2. On the Home page, access **Setting and Actions** by clicking your user name.
3. Click **Downloads**.
4. In the Downloads page, click **Download for Windows** in the EPM Automate section.
5. Save the installer to your computer.
6. Right-click the installer (`EPM Automate.exe`), and select `Run as administrator`.
7. In **User Account Control**, click **Yes**.
8. Follow on-screen prompts to complete the installation.

Linux/UNIX/macOS X Instructions

EPM Automate requires access to a deployment of a supported JRE (version 8 through 11). The environment variable `JAVA_HOME` must be set to point to your JRE installation.

To install EPM Automate:

1. Access an environment.
2. On the Home page, access **Setting and Actions** by clicking your user name.
3. Click **Downloads**.
4. In the Downloads page, click **Download for Linux/macOS X** in the EPM Automate section.
5. Save the installer (`EPMAutomate.tar`) in a directory in which you have read/write/execute privileges.
6. Extract the contents of the installer, set the required environment variables and execute `epmautomate.sh`:

   macOS X example (Bash shell assumed) to install and run from your home directory.

   ```
   cd ~/
   tar xf path_to_downloaded_EPMAutomate.tar
   export JAVA_HOME=$(/usr/libexec/java_home)
   export PATH $HOME/epmautomate/bin:$PATH
   epmautomate.sh
   ```

   Linux example (Bash shell assumed) to install and run from your home directory. JDK version 1.8.0_191 is assumed.

   ```
   cd ~/
   tar xf path_to_downloaded_EPMAutomate.tar
   export JAVA_HOME=/opt/jdk1.8.0_191
   ```

```
export PATH ~/Downloads/epmautomate/bin:$PATH
epmautomate.sh
```

Server-side Execution of EPM Automate Commands

Some EPM Automate commands can be run directly in Oracle Enterprise Performance Management Cloud using Groovy. You do not need to install EPM Automate to run commands using Groovy scripts.

Note that server-side execution of commands is not the same as running Groovy scripts on a client computer to execute EPM Automate commands.

For detailed information, see Running Commands without Installing EPM Automate.

Understanding EPM Automate Encryption Level

Oracle Enterprise Performance Management Cloud uses Transport Layer Security (TLS) with SHA-2/SHA-256 Cryptographic Hash Algorithm to secure communication with EPM Automate.

Using OAuth 2.0 Authorization Protocol with OCI (Gen2) Environments

EPM Automate can use the OAuth 2.0 authentication protocol to access OCI (GEN 2) Oracle Enterprise Performance Management Cloud environments to execute commands, especially for automating the running of commands.

To enable OAuth 2.0 access, an Identity Domain Administrator must register your application as a public client in Oracle Cloud Identity Services. OAuth is enforced for the application; not across your subscription.

For detailed instructions on setting up OAuth 2.0 for your OCI (Gen 2) environments, see Authentication with OAuth 2 - Only for OCI (Gen 2) Environments in *REST API for Oracle Enterprise Performance Management Cloud*.

> **Note:**
>
> Basic authentication works even when OAuth is enabled for an environment. Be sure to not overwrite the existing encrypted password file if you plan to use it in the future.

Creating an Encrypted Password File Containing Refresh Token and Client ID

Service Administrators who want to use OAuth 2.0 for EPM Automate access to environments require these details to create their encrypted password file, which is then used to sign into the environment:

- Refresh token
  See steps under "EPM Cloud Service Administrator tasks to get a refresh token:" in Authentication with OAuth 2 - Only for OCI (Gen 2) Environments in *REST API for Oracle Enterprise Performance Management Cloud* for detailed instructions on how to get the refresh token.
- Client ID

The Client ID is generated when the Identity Domain Administrator configures the application for OAuth. It is visible on the Configuration tab of the application, under **General Information**.

To create the encrypted password file for OAuth authentication:

1. Start an EPM Automate session.
2. Execute a command similar to the following:
   `epmautomate encrypt` *`REFRESH_TOKEN ENCRYPTION_KEY PASSWORD_FILE`* `ClientID=`*`CLIENT_ID`*, where, the *`REFRESH_TOKEN`* is the decrypted refresh token from the secure store and *`ENCRYPTION_KEY`* is any private key to encrypt the password, and *`PASSWORD_FILE`* is the name and location of the file that stores the encrypted refresh token. IThe password file must use the `.epw` extension.

   See encrypt for detailed instructions.
3. Use the newly generated password file to sign in using OAuth. For automated script executions, be sure to update scripts to point to the newly generated password file.

2

Command Reference

Some EPM Automate commands apply to all business processes while some apply to a group of business processes. Unless otherwise specified, a command applicable to a specific business process (for example Planning) does not work with a different business process (for example, Financial Consolidation and Close). Attempts to execute a command against a business process that does not support it will result in an error. For a list of commands applicable to each business process, see Commands Specific to Each EPM Cloud Service.

About Running EPM Automate Commands

All Oracle Enterprise Performance Management Cloud services use EPM Automate commands for remote administration of environments.

Prerequisites

This section lists the prerequisites for using EPM Automate such as the use of Oracle Enterprise Performance Management Cloud credentials and default file locations in environments.

General

All EPM Cloud users can use their identity domain credentials to connect to an environment using EPM Automate. The predefined roles and application roles assigned to the user decide the commands that a user can execute.

- Additionally, the Identity Domain Administrator role is required to run commands that add or delete users in the identity domain.
- Any file required to execute a command must exist within the environment. You use the uploadFile command to upload files.

  See Default File Locations for information on the default file location used by each service.
- File extension usage in commands:

  - Specify the full file name, including the file extension (for example, data.csv), to run commands that perform file operations. Example of file operation commands include `deletefile` `listfiles`, `uploadfile`.
  - Do not use file extensions to run commands that perform Migration operations. Migration operations require you to specify the name of a snapshot.
- Parameter values that contain a space character; for example, comments, location names and folder paths, must be enclosed in quotation marks.

Planning

- Jobs
  Many of the commands discussed in the following section require jobs. 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, and refreshing the database.

  Using the Jobs Console, you must create appropriate jobs to perform the following operations. For detailed instructions on creating jobs in Planning, see "Managing Jobs" in *Administering Planning*.
  - Import data into an application
  - Export data from an application
  - Import metadata into an application
  - Export metadata from an application
  - Copy data from one a block storage database to an aggregate storage database or from a block storage database to another block storage database
- Business Rules
  Business rules that you want to execute must exist in the application.

  You use Calculation Manager to create business rules, which are then deployed into the application. See *Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud*.

Data Management

- Data Rules
  Data load rules define how Data Management loads data from a file. You must have predefined data load rules to load data using EPM Automate.
- Batches
  You can load data using batches defined in Data Management. Using a batch, users can combine many load rules in a batch and execute them in serial or parallel mode.

Default File Locations

Default Upload Location

By default, all uploaded files to Oracle Enterprise Performance Management Cloud is stored in a default location that is accessible to Migration.

You must upload files that are to be processed by Migration, for example, snapshots that you want to import into the service, to the default location.

Inbox and Outbox

The inbox and outbox locations may differ across EPM Cloud business processes. You use the inbox to upload files that you want to import or otherwise process using a business processes other than Profitability and Cost Management. Data Management can process files in the inbox or a directory within it.

Typically, EPM Cloud stores files that you generate through the business processes, for example, data or metadata export files, in the outbox.

- The inbox to which EPM Automate uploads files and the outbox that stores the files for download is accessible to these applications. You must upload files to this location if you plan to process them using a process that is native to these applications. You may also uplod files to the outbox.
  - Planning
  - Planning Modules
  - Account Reconciliation
  - Financial Consolidation and Close
  - Tax Reporting
  - Narrative Reporting
  - Enterprise Profitability and Cost Management

  You can use the Inbox/Outbox Explorer to browse the files stored in the default location. Application snapshots that you create using EPM Automate are not listed in the Inbox/Outbox Explorer; you can view them from the Snapshots tab of Migration.
- Files that are to be processed using a Profitability and Cost Management processes must be uploaded into `profitinbox`. You may also upload files to the profitoutbox. Files exported by Profitability and Cost Management processes are stored in `profitinbox`. You use the File Explorer to browse these files.
- Files that are to be processed using Data Management must be available in the inbox or in a folder within it. By default, files exported using Data Management are stored in Outbox while Data Management report outputs are stored in Data Management `outbox/report` folder. You use the Data Management File Browser to browse these files.
- Oracle Enterprise Data Management Cloud uses the default location for import and export files which are uploaded, copied, or downloaded. Files in the default location can be viewed using the `ListFiles` command.

Log Files

Each EPM Automate command execution generates a debug file, which is automatically deleted if the command is successful. If an error occurs during command execution, the debug file for the failed command is maintained in the directory from which you run EPM Automate. By default, this is the `Oracle/epm automate/bin` directory (Windows) or `home/user/epmautomate/bin` (Linux/UNIX).

EPM Automate debug files use the following naming convention:

`commandname_date_timestamp.log`. For example, if you run a failed `listfiles` command at 09:28:02 on November 23, 2020, the debug file name is `listfiles_23_11_2020_09_28_02.log`.

You cannot suppress the creation of the debug file for a failed command. You can, however, write debug information and command output to a file in a different directory by appending `-d` along with a debug file name and error and output streams (`-d >> c:\logs\LOG_FILE_NAME.log 2>&1`) to end of the command as shown in the following Windows example:

```
epmautomate listfiles -d >> c:\logs\listfiles.log 2>&1
```

Enable Transport Layer Security Protocol 1.2

EPM Automate must be installed on an operating system that supports Transport Layer Security (TLS) protocol 1.2 or higher.

To ensure the highest level of security for authentication and data encryption, EPM Automate supports only TLS 1.2. If TLS 1.2 is not enabled on the computer from which EPM Automate is run, `EPMAT-7: Unable to connect. Unsupported Protocol: HTTPS` error is displayed. To resolve this error, work with your IT administrator to enable TLS 1.2.

The procedures to enable TLS 1.2 is operating system dependent. Use these information sources; similar web resources may be available for other supported operating systems:

- Update to enable TLS 1.1 and TLS 1.2 as default secure protocols in WinHTTP in Windows for information on enabling TLS 1.2 for Windows computers.
- Hardening TLS Configuration for information on enabling TLS 1.2 in OpenSSL for Red Hat Enterprise Linux.

Using EPM Automate Commands

Sequence of Command Parameters

All mandatory parameters for a command must be passed in the sequence identified in command usage. Mandatory parameters and their values precede optional parameters, which can be passed in any sequence. Optional parameters are not positional.

For example, consider the following usage of the `login` command:

```
epmautomate login USERNAME PASSWORD EPM-CLOUD_BASE_URL
[ProxyServerUserName=PROXY_USERNAME] [ProxyServerPassword=PROXY_PASSWORD]
[ProxyServerDomain=PROXY_DOMAIN]
```

This command has three mandatory parameters; `USERNAME`, `PASSWORD`, and `EPM-CLOUD_BASE_URL`, which should appear in the sequence identified in the usage. The command will return an error if this sequence is not maintained. Optional parameters `ProxyServerUserName`, `ProxyServerPassword`, and `ProxyServerDomain` and their values can be specified in any sequence.

Are EPM Automate Commands Case-Sensitive?

EPM Automate commands are not case-sensitive. How a command name is typed has no impact on command execution. For example, you can type the `addUsers` command as `addusers`, `ADDUSERS`, or `AdDuSeRs`.

Are EPM Automate Command Parameters Case-Sensitive?

EPM Automate command parameters are not case-sensitive. How a command parameter name is typed has no impact on command execution. For example, you can type the `FileName` parameter as `filename`, `fileName`, or `fIlEnAmE` without impacting command execution.

Specifying Multiple Values for a Parameter

Some EPM Automate commands accept multiple comma separated parameter values; for example, a run time prompt of type members in business rules, rulesets and templates in a Planning application.

To set more than one member for a Members type of run time prompt named `Entities` in an EPM Automate command, use a , (comma) as illustrated in the following example for executing the `runbusinessrule` command.

```
epmautomate runbusinessrule clearDistData TargetYear=FY19
TargetMonth=Feb Entities=District1,District2
```

Member names containing special characters such as Space and Comma must be enclosed in quotation marks and escaped using \ (backslash) as shown in the following example:

```
epmautomate runbusinessrule clearDistData TargetYear=FY19
TargetMonth=Feb Entities="\"District 1\",\"entity_name, with comma\""
```

Behavior During Daily Maintenance

Do not run EPM Automate commands while the daily maintenance of an environment is in progress.

User activity is not permitted during daily maintenance. Attempts to run EPM Automate commands, either directly or using scripts, while daily maintenance is in process will display the following error:

```
EPMAT-11:Internal server error. Due to the daily maintenance, your
Oracle EPM Cloud Service environment is currently unavailable.
```

Running EPM Automate

You use your Oracle Enterprise Performance Management Cloud credentials to sign in using EPM Automate. You cannot sign in using your SSO credentials.

All EPM Cloud users can use their identity domain credentials to connect to an environment using EPM Automate. The predefined and application roles assigned to the user determine the commands that a user can execute.

Additionally, only Service Administrators can run some commands while Identity Domain Administrator roles may also be required to run some commands.

Generating Debug Log File

Oracle Support will ask you for a debug log file of the session to troubleshoot problems that you encountered while running EPM Automate. EPM Automate supports the `-d` option to generate debug messages, which can then be redirected to a file using `>` directive. You can create a debug file for one command or a batch execution file or script containing several commands.

Usage: `epmautomate command [command_parameters] -d > log_file 2>&1`

Windows Example: `epmautomate downloadfile "Artifact Snapshot" -d > C:\logs\download_log.txt 2>&1`

Linux Example: `epmautomate.sh downloadfile "Artifact Snapshot" -d > ./logs/download_log 2>&1`

Windows

Before running EPM Automate, ensure that you can access your environment from the computer from which you are running EPM Automate.

EPM Automate creates a `.prefs` file, which contains user information, and log files in the current directory. On Windows computers, the contents of `.prefs` file is visible only to the user who created it and to Windows administrators. In Linux, UNIX and macOSX environments, `.prefs` file is generated with permission 600, which allows only the owner to read and write to this file.

EPM Automate displays `FileNotFoundException: .prefs (Access is denied)` error in Windows environments if you do not have write permission in the Windows directory from which you execute EPM Automate. To resolve this error, ensure that the Windows account of the current user has Read/Write access to the directory from which EPM Automate is run. Additionally, this user must have appropriate access to any other directory from which a file is accessed (for example, while running the `uploadFile` command) or written (for example, while running the `downloadFile` command).

> **Note:**
>
> You cannot run EPM Automate from a folder that contains `&` in its name; for example, `C:\Oracle\A&B`.

To run EPM Automate on a Windows client:

1. Click **Start**, then **All Programs**, then **EPM Automate**, and then **Launch EPM Automate**. The EPM Automate command prompt is displayed.
2. **Optional:** Navigate to the directory from which you want to perform operations using EPM Automate.
3. **Optional:** Generate a password encryption file. You use the password encryption file to pass encrypted password to initiate a session.

   ```
   epmautomate encrypt P@ssword1 myKey C:/mySecuredir/password.epw
   ```

4. Start a session as a Service Administrator. Use a command such as the following:
   - Using an unencrypted password:

     ```
     epmautomate login serviceAdmin P@ssword1
     https://test-cloudpln.pbcs_us1.oraclecloud.com
     ```

- Using an encrypted password:

```
epmautomate login serviceAdmin C:\mySecuredir\password.epw
https://test-cloudpln.pbcs_us1.oraclecloud.com
```

5. Enter commands to execute the tasks you want to complete.

   See Exit Codes for information on command execution status.

6. Sign out of the environment. Use the following command:

```
epmautomate logout
```

Linux

> **Note:**
>
> Ensure that *JAVA_HOME* is set in the `PATH` variable of your `.profile` file or as a shell environment variable. A supported JRE (version 8 through 11) is required.

To run EPM Automate on a Linux client:

1. Open a terminal window and navigate to the directory where you installed EPM Automate.
2. **Optional:** Generate a password encryption file. You use the password encryption file to pass an encrypted password instead of an unencrypted password to initiate a session.

```
epmautomate encrypt P@ssword1 myKey ../misc/encrypt/password.epw
```

3. Start a session as a Service Administrator. Use a command such as the following:
   - Using an unencrypted password:

```
./bin/epmautomate.sh login serviceAdmin P@ssword1
https://test-cloudpln.pbcs_us1.oraclecloud.com
```

   - Using an encrypted password:

```
./bin/epmautomate.sh login serviceAdmin ../misc/encrypt/password.epw
https://test-cloudpln.pbcs_us1.oraclecloud.com
```

4. Enter commands to execute the tasks you want to complete.

   See Exit Codes for information on command execution status.

5. Sign out of the environment. Use the following command:

```
./bin/epmautomate.sh logout
```

Running Multiple Instances of EPM Automate

You can run multiple instances of EPM Automate against one environment from the same directory. Similarly, you can run multiple instances of EPM Automate against different environments from the same or different directories.
For example, you may need to simultaneously refresh the Planning application cube in `https://cloudpln.pbcs.us1.oraclecloud.com` and `https://testcloudpln.pbcs.us1.oraclecloud.com`. In this scenario, you have two options:

- Run two instances of EPM Automate from the same directory to refresh application cubes in different environments
- Execute EPM Automate from separate directories to connect to the environments and then refresh application cubes

In both scenarios, each instance of EPM Automate works independently; logging out of one instance does not log you out of other instances. Activities initiated using EPM Automate continues to run to completion in the environment even if you sign out from the other instance.

This section contains Windows and Unix/Linux sample scripts (`caller` and `multisession`) that may be used to create two EPM Automate sessions to perform tasks. To run multiple simultaneous sessions, you must add the following connection information in the `caller` script, which calls the `multisession` script to run `login`, `uploadfile`, `listfiles`, and `logout` commands. You can modify the `multisession` script to perform tasks other than these. Make sure that both these scripts are stored in the same directory.

- EPM Automate uses the environment variable `EPM_SID` to distinguish multiple sessions. This variable must be set in the caller script to a unique value for each session. In the sample scripts, it is set to unique values as follows:
  - In `caller.BAT`, `EPM_SID` is set to `!RANDOM!`, which assigns it a unique system generated number. This number is also used to generate the log files for each session. If you want to track the log file for each session, you may specify a unique number instead of `!RANDOM!`.
  - In `caller.sh`, `EPM_SID` is set to the process ID, which is unique. If you want to track the log file for each session, you may specify a unique `EPM_SID` by modifying the `export EPM_SID=$$` statement in the `multisession` script to use the passed in value, and then pass a unique value for this parameter in the `caller` script for each session, for example by specifying the value of `EPM SID` in `caller.sh` as follows:

    ```
    $SCRIPT_DIR/multisession.sh EPM_SID "USERNAME" "PASSWORD" "URL" "/home/
    user/Snapshot1.zip" &
    $SCRIPT_DIR/multisession.sh EPM_SID "USERNAME" "PASSWORD" "URL" "/home/
    user/Snapshot2.zip" &
    ```

- `USERNAME`: Login ID of a Service Administrator
- `PASSWORD`: Password of the Service Administrator
- `URL`: Connection URL of the environment

Sample Windows Scripts

caller.BAT

```
@echo off
setlocal EnableExtensions EnableDelayedExpansion
```

```
REM syntax: start /B multisession.bat EPM_SID "USERNAME" "PASSWORD" "URL"
"SNAPSHOTPATH"
start /B multisession.bat !RANDOM! "USERNAME" "PASSWORD" "URL"
"C:\Snapshot1.zip"
start /B multisession.bat !RANDOM! "USERNAME" "PASSWORD" "URL"
"C:\Snapshot2.zip"

endlocal
```

multisession.BAT

```
@echo off

set EPM_SID=%1
set USERNAME=%2
set PASSWORD=%3
set URL=%4
set SNAPSHOTNAME=%5

echo User: %USERNAME% > %EPM_SID%.log
echo Cloud Instance: %URL% >> %EPM_SID%.log

call epmautomate login %USERNAME% %PASSWORD% %URL% >> %EPM_SID%.log
call epmautomate uploadfile %SNAPSHOTNAME% >> %EPM_SID%.log
call epmautomate listfiles >> %EPM_SID%.log
call epmautomate logout
```

Sample Bourne Shell Script

caller.sh

```
#!/bin/sh

set +x
SCRIPT_DIR=`dirname "${0}"`

# syntax: /home/user/multisession.sh "USERNAME" "PASSWORD" "URL"
"SNAPSHOTPATH" &
$SCRIPT_DIR/multisession.sh "USERNAME" "PASSWORD" "URL"  "/home/user/
Snapshot1.zip" &
$SCRIPT_DIR/multisession.sh "USERNAME" "PASSWORD" "URL" "/home/user/
Snapshot2.zip" &
```

multisession.sh

```
#!/bin/sh

set +x

EPM_AUTOMATE_HOME=/home/user/epmautomate

export JAVA_HOME=/home/user/jre
export EPM_SID=$$
```

```
USERNAME=$1
PASSWORD=$2
URL=$3
SNAPSHOTNAME=$4

echo User: $USERNAME > $EPM_SID.log
echo Cloud Instance: $URL >> $EPM_SID.log

$EPM_AUTOMATE_HOME/bin/epmautomate.sh login $USERNAME $PASSWORD $URL
>> $EPM_SID.log
$EPM_AUTOMATE_HOME/bin/epmautomate.sh uploadfile $SNAPSHOTNAME >> $EPM_SID.log
$EPM_AUTOMATE_HOME/bin/epmautomate.sh listfiles >> $EPM_SID.log
$EPM_AUTOMATE_HOME/bin/epmautomate.sh logout
```

Commands at a Glance

This is an alphabetical listing of all EPM Automate commands.

Table 2-1 All EPM Automate Commands

Command Name	PLN, SWP, SP, FF	FCC	TR	PCM	EPCM	AR	EDM	NR
addUsers	✓	✓	✓	✓	✓	✓	✓	✓
addUsersToGroup	✓	✓	✓	✓	✓	✓	✓	✓
addUsersToTeam		✓	✓			✓		
addUserToGroups	✓	✓	✓	✓	✓	✓	✓	✓
applicationAdminMode	✓	✓	✓		✓			
applyDataGrants				✓				
archiveTmTransactions						✓		
assignRole	✓	✓	✓	✓	✓	✓	✓	✓
autoPredict* See Footnote	✓							
calculateModel					✓			
clearCube	✓				✓			
clearDataByPointOfView					✓			
clearDataByProfile		✓	✓					
clearPOV				✓				
cloneEnvironment	✓	✓	✓	✓	✓	✓	✓	✓
copyDataByPointOfView					✓			
copyDataByProfile		✓	✓					
copyFileFromInstance	✓	✓	✓	✓	✓	✓	✓	✓
copyFromObjectStorage	✓	✓	✓	✓	✓	✓	✓	✓
copyOwnershipDataToNextYear		✓	✓					
copyPOV				✓				
copySnapshotFromInstance	✓	✓	✓	✓	✓	✓	✓	
copyToObjectStorage	✓	✓	✓	✓	✓	✓	✓	✓
createGroups	✓	✓	✓	✓	✓	✓	✓	✓

Table 2-1 (Cont.) All EPM Automate Commands

Command Name	PLN, SWP, SP, FF	FCC	TR	PCM	EPCM	AR	EDM	NR
createNRSnapshot								✓
createReconciliations						✓		
deleteFile	✓	✓	✓	✓	✓	✓	✓	✓
deleteGroups	✓	✓	✓	✓	✓	✓	✓	✓
deletePointOfView					✓			
deletePOV				✓				
deployCube				✓				
deployEJTemplates		✓						
deployFormTemplates		✓	✓					
deployTaskManagerTemplate		✓						
dismissIPMInsights**	✓							
downloadFile	✓	✓	✓	✓	✓	✓	✓	✓
enableApp				✓				
enableQueryTracking	✓				✓			
encrypt	✓	✓	✓	✓	✓	✓	✓	✓
essbaseBlockAnalysisReport	✓	✓	✓					
executeAggregationProcess	✓				✓			
executeBurstDefinition								✓
executeReportBurstingDefinition	✓	✓	✓		✓			
exportAccessControl						✓		
exportAppAudit	✓	✓	✓		✓			
exportAppSecurity	✓	✓	✓		✓			
exportARApplicationProperties						✓		
exportBackgroundImage						✓		
exportCellLevelSecurity	✓		✓		✓			
exportData	✓	✓	✓		✓			
exportConsolidationJournals		✓						
exportDataManagement	✓	✓	✓	✓	✓			
exportDimension							✓	
exportDimensionMapping							✓	
exportEJJournals		✓						
exportEssbaseData	✓	✓	✓		✓			
exportJobConsole	✓	✓	✓		✓			
exportLibraryArtifact								✓
exportLibraryDocument	✓	✓	✓		✓			
exportLogoImage						✓		
exportMapping	✓	✓	✓	✓	✓	✓		
exportMetadata	✓	✓	✓		✓			
exportOwnershipData		✓	✓					
exportQueryResults				✓				

Table 2-1 (Cont.) All EPM Automate Commands

Command Name	PLN, SWP, SP, FF	FCC	TR	PCM	EPCM	AR	EDM	NR
exportSnapshot	✓	✓	✓	✓	✓	✓	✓	
exportTemplate				✓				
exportTaskManagerAccessControl		✓	✓					
exportValidIntersections	✓	✓	✓		✓			
extractDimension							✓	
extractPackage							✓	
feedback	✓	✓	✓	✓	✓	✓	✓	✓
getApplicationAdminMode	✓	✓	✓		✓	✓		
getDailyMaintenanceStartTime	✓	✓	✓	✓	✓	✓	✓	✓
getEssbaseQryGovExecTime	✓	✓	✓	✓	✓			
getIdleSessionTimeout	✓	✓	✓	✓	✓	✓	✓	✓
getIPAllowlist	✓	✓	✓	✓	✓	✓	✓	✓
getRestrictedDataAccess	✓	✓	✓	✓	✓	✓	✓	✓
getSubstVar	✓	✓	✓		✓			
getVirusScanOnFileUploads	✓	✓	✓	✓	✓	✓	✓	✓
groupAssignmentAuditReport	✓	✓	✓	✓	✓	✓	✓	✓
help	✓	✓	✓	✓	✓	✓	✓	✓
importAppAudit	✓				✓			
importAppSecurity	✓	✓	✓		✓			
importARApplicationProperties						✓		
importBackgroundImage						✓		
importBalances						✓		
importCellLevelSecurity	✓		✓		✓			
importConsolidationJournals		✓						
importData	✓	✓	✓	✓	✓			
importDataManagement	✓	✓	✓	✓	✓			
importDimension							✓	
importJobConsole	✓	✓	✓		✓			
importLibraryArtifact								✓
importLogoImage						✓		
importMapping	✓	✓	✓	✓	✓	✓		
importMetadata	✓	✓	✓		✓			
importOwnershipData		✓	✓					
importPreMappedBalances						✓		
importPreMappedTransactions						✓		
importProfiles						✓		
importRates						✓		
importRCAttributeValues						✓		
importReconciliationAttributes						✓		
importSnapshot	✓	✓	✓	✓	✓	✓	✓	

Table 2-1 (Cont.) All EPM Automate Commands

Command Name	PLN, SWP, SP, FF	FCC	TR	PCM	EPCM	AR	EDM	NR
importSupplementalCollectionData		✓	✓					
importSupplementalData		✓	✓					
importTemplate				✓				
importTMAttributeValues						✓		
importValidIntersections	✓	✓	✓		✓			
invalidLoginReport	✓	✓	✓	✓	✓	✓	✓	✓
listBackups	✓	✓	✓	✓	✓	✓	✓	✓
listFiles	✓	✓	✓	✓	✓	✓	✓	✓
loadData				✓				
loadDimData				✓				
loadViewpoint							✓	
login	✓	✓	✓	✓	✓	✓	✓	✓
logout	✓	✓	✓	✓	✓	✓	✓	✓
maskData	✓	✓	✓		✓			
mergeDataSlices	✓				✓			
mergeSlices				✓				
optimizeASOCube				✓				
programDocumentationReport				✓				
provisionReport	✓	✓	✓	✓	✓	✓	✓	✓
purgeArchivedTmTransactions						✓		
purgeTmTransactions						✓		
recomputeOwnershipData		✓	✓					
recreate	✓	✓	✓	✓	✓	✓	✓	✓
refreshCube	✓	✓	✓		✓			
removeUserFromGroups	✓	✓	✓	✓	✓	✓	✓	✓
removeUsers	✓	✓	✓	✓	✓	✓	✓	✓
removeUsersFromGroup	✓	✓	✓	✓	✓	✓	✓	✓
removeUsersFromTeam		✓	✓			✓		
renameSnapshot	✓	✓	✓	✓	✓	✓	✓	
replay	✓	✓	✓	✓	✓	✓	✓	✓
resetService	✓	✓	✓	✓	✓	✓	✓	✓
restoreBackup	✓	✓	✓	✓	✓	✓	✓	✓
restructureCube	✓	✓	✓					
roleAssignmentAuditReport	✓	✓	✓	✓	✓	✓	✓	✓
roleAssignmentReport	✓	✓	✓	✓	✓	✓	✓	✓
runAutomatch						✓		
runBatch	✓	✓	✓	✓	✓	✓		
runBusinessRule	✓	✓	✓					
runCalc				✓				
runComplianceReport						✓		

Table 2-1 (Cont.) All EPM Automate Commands

Command Name	PLN, SWP, SP, FF	FCC	TR	PCM	EPCM	AR	EDM	NR
runDailyMaintenance	✓	✓	✓	✓	✓	✓	✓	✓
runDataRule	✓	✓	✓	✓	✓	✓		
runDMReport	✓	✓	✓	✓	✓	✓		
runIntegration	✓	✓	✓	✓	✓	✓		
runIntercompanyMatchingReport		✓						
runMatchingReport						✓		
runPlanTypeMap	✓							
runRuleSet	✓	✓	✓					
runSupplementalDataReport		✓	✓					
runTaskManagerReport		✓	✓					
sendMail	✓	✓	✓	✓	✓	✓	✓	✓
setApplicationAdminMode	✓	✓	✓		✓	✓		
setDailyMaintenanceStartTime	✓	✓	✓	✓	✓	✓	✓	✓
setDemoDates		✓	✓			✓		
setEJJournalStatus		✓						
setEncryptionKey	✓	✓	✓	✓	✓	✓	✓	✓
setEssbaseQryGovExecTime	✓	✓	✓	✓	✓			
setIdleSessionTimeout	✓	✓	✓	✓	✓	✓	✓	✓
setIPAllowlist	✓	✓	✓	✓	✓	✓	✓	✓
setManualDataAccess	✓	✓	✓	✓	✓	✓	✓	✓
setPeriodStatus						✓		
setRestrictedDataAccess	✓	✓	✓	✓	✓	✓	✓	✓
setSubstVars	✓	✓	✓		✓			
setVirusScanOnFileUploads	✓	✓	✓	✓	✓	✓	✓	✓
simulateConcurrentUsage	✓	✓	✓					
skipUpdate	✓	✓	✓	✓	✓	✓	✓	✓
snapshotCompareReport	✓	✓	✓		✓			
sortMember	✓				✓			
unassignRole	✓	✓	✓	✓	✓	✓	✓	✓
updateUsers	✓	✓	✓	✓	✓	✓	✓	✓
upgrade	✓	✓	✓	✓	✓	✓	✓	✓
uploadFile	✓	✓	✓	✓	✓	✓	✓	✓
userAuditReport	✓	✓	✓	✓	✓	✓	✓	✓
userGroupReport	✓	✓	✓	✓	✓	✓	✓	✓
validateConsolidationMetadata		✓						
validateModel					✓			

- * This command is supported only if Hybrid Oracle Essbase cubes are enabled in the application. Strategic Workforce Planning and Sales Planning do not support Hybrid Essbase. This command is not supported for FreeForm.
- ** This command is not supported for FreeForm.

Abbreviations

- PLN: Planning (including Planning Modules)
- FF: FreeForm
- SWP: Strategic Workforce Planning
- SP: Sales Planning
- FCC: Financial Consolidation and Close
- TR: Tax Reporting
- PCM: Profitability and Cost Management
- EPCM: Enterprise Profitability and Cost Management
- AR: Account Reconciliation
- EDM: Oracle Enterprise Data Management Cloud
- NR: Narrative Reporting

EPM Automate Commands

This section details each EPM Automate command. Information available for each command includes the services that can use the command, command usage, and example.

addUsers

Creates a batch of users in an identity domain using an ANSI or UTF-8 encoded Comma Separated Value (CSV) file that was uploaded to the environment. Also informs new users of their user name and temporary password.

You use the uploadFile command to upload files to an environment. All columns in the CSV file are mandatory. This command validates the value in each column of a definition and displays error messages that identify each missing or invalid value. The CSV 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
```

See Importing a Batch of User Accounts in *Getting Started with Oracle Cloud* for a detailed description of the CSV file format.

The value of `User Login` specified in the import file is not case-sensitive. For example, the value `John.doe@example.com` is treated as being identical to `John.Doe@example.com` or any variation in its case.

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 command creates accounts only for new users whose account information is included in the file. Because user accounts are common to all environments that an identity domain supports, new users are available to all the environments that share the identity domain.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

Usage

`epmautomate addUsers` *`FILE_NAME`* `[userPassword=`*`PASSWORD`*`] [resetPassword=true|false]`
where:

- *`FILE_NAME`* is the name of a CSV file containing user information. Input file containing multi-byte characters must use UTF-8 character encoding. Using ANSI encoding causes issues in how user information is displayed in My Services screens.
- `userPassword`, optionally, indicates the default password for all the new users who are created in the identity domain. If specified, this password must meet the minimum identity domain password requirements. If the parameter is not specified, a unique temporary password is assigned to each user.
  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 Oracle Enterprise Performance Management Cloud users and want to assign a specific password to each user, use this command without specifying a value for the `userPassword` optional parameter.
- `resetPassword`, optionally, indicates whether new users must change password at the first log in. Default is `true`. Unless this parameter is set to `false`, new users will be forced to change the password at the first sign in.
  This command 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 must 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 and they will not be able to login.

Examples

- Add test users in the identity domain with the same password and not require them to change password:
  ```
  epmautomate addUsers user_file.CSV userPassword=Example@Pwd12
  resetPassword=false
  ```
- Add users to identity domain using a temporary password and require them to change it:
  ```
  epmautomate addUsers user_file.CSV
  ```

addUsersToGroup

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.

You use the uploadFile command to upload files to an environment. User login value is not case-sensitive. The file format is as follows:

```
User Login
jdoe
john.doe@example.com
```

> **Note:**
>
> User is added to group only if both these conditions are met:
>
> - User Login value included in the file exists in the identity domain that services the environment. User Login values are not case-sensitive.
> - The user is assigned to a predefined role in the identity domain

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Enterprise Profitability and Cost Management, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate addUsersToGroup` *`FILE_NAME GROUP_NAME`* where:

- *`FILE_NAME`* is the name of a CSV file containing the login names of users you want to assign to a group in Access Control.
- *`GROUP_NAME`* is the name of a group existing in Access Control. This value is not case-sensitive.

Example

```
epmautomate addUsersToGroup user_file.CSV example_group
```

addUsersToTeam

Adds Oracle Enterprise Performance Management Cloud users listed in a CSV file to an existing team.

If a user included in the CSV file is already a member of the team, this command ignores that user. The values in this file are nto case-sensitive. The CSV file format is as follows:

```
User Login, primary_user
jdoe, yes
jane.doe@example.com,no
```

> **Note:**
>
> A primary user is, by default, designated to perform the tasks that are assigned to the team.

Applies to

Financial Consolidation and Close, Tax Reporting, and Account Reconciliation.

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate addUsersToTeam` *`FILE TEAM_NAME`* where:

- `FILE` identifies a UTF-8 formatted CSV file listing the login IDs of users to add to the team. Before running this command, use the uploadFile command to upload files to an environment.
- `TEAM_NAME` identifies a team name as defined in Access Control. This value is not case-sensitive.

Example

```
epmautomate addUsersToTeam example_users.csv example_team
```

addUserToGroups

Adds a user as a member of the Access Control groups identified in an ANSI or UTF-8 encoded CSV file.

You use the uploadFile command to upload files to an environment. The file format is as follows:

```
Group Name
Group1
Group2
```

Group Name values are not case-sensitive.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate addUserToGroups` *`FILE_NAME User_Login`*, where:

- `FILE_NAME` is the name of a CSV file containing the Access Control group names to which you want to assign to assign the user
- `User_Login` is the log in ID of an Oracle Enterprise Performance Management Cloud user who is to be assigned to Access Control groups. This user login ID, which is not case-sensitive, must exist in the identity domain that services the environment and must be assigned to a predefined role.

Example

```
epmautomate addUserToGroups groups.CSV jdoe@examle.com
```

applicationAdminMode

Places the application in administration mode so that access to the application is limited to Service Administrators only.

This command is useful to prevent users from working on the application when Service Administrators are performing administrative operations. The application remains in administration mode until you change it back so that all users can access it.

> **Note:**
>
> This command has been deprecated, but not removed from EPM Automate. Oracle recommends that you use the setApplicationAdminMode command instead.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate applicationAdminMode` *`VALUE`* where `VALUE` specifies whether to place the application in administration mode. Acceptable values are:

- `true` to place the application in administration mode
- `false` to return the application to normal mode so that all users can access it

Examples

- Place the application in administration mode: `epmautomate applicationAdminMode true`
- Return the application to normal operation: `epmautomate applicationAdminMode false`

applyDataGrants

Refreshes the data grants, which control access to Oracle Essbase data slices, so that they match the data grants defined in an Profitability and Cost Management application.

User and group level data grants that you make in the Profitability and Cost Management application are automatically synchronized in Essbase. Use this command to synchronize access to Essbase data if you suspect a discordance between the data grant in the application and the filters in Essbase.

The time required to complete this operation depends on the size of the application. Make sure that the data grant refresh operation finishes before the application is backed up during the next maintenance window. Because the application should not be used while this operation is in progress, Oracle recommends that you schedule this operation for a time when users are not working with the application.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate applyDataGrants` *`APPLICATION_NAME`* where *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application for which data grants are to be re-created.

Example

```
epmautomate applyDataGrants BksML12
```

archiveTmTransactions

Archives matched transactions, including support and adjustment details, that are equal to or older than a specified age. The matched transactions are recorded in a ZIP file.

Use this command to keep the Account Reconciliation application size optimal by archiving and then purging old matched transactions based on the transaction retention policies of your organization.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate archiveTmTransactions matchType age [filterOperator=VALUE] [filterValue=VALUE] [logFilename=FILE_NAME] [filename=FILE_NAME]` where:

- `matchType` is the identifier (`TextID`) of the match type from which matched transactions should be archived.
- `age` identifies the number of days since the transaction was matched. Matched transaction older than or equal to this value will be archived.
- `filterOperator`, optionally, is one of the following filter conditions to identify the accounts containing matched transactions for archival. This value is combined with the `filterValue` to identify the accounts from which matched transactions should be archived:
  - `equals`
  - `not_equals`
  - `starts_with`
  - `ends_with`
  - `contains`
  - `not_contains`
- `filterValue`, optionally, is a filter value to identify the transactions to archive. If the `filterOperator` is `equals` or `not_equals`, you can use a space-separated list to specify multiple values; for example, `filterValue=101-120 filterValue=102-202`. If multiple values are specified, transactions from accounts matching any filter operator and filter value combination are selected for archival.

  > **Note:**
  >
  > If `filterOperator` and `filterValue` are not specified, all matched transactions older than or equal to the `age` from all accounts for the specified `matchType` are archived.

- `logFilename`, optionally, is the name of a log file to record information about the command activity. If a file name is not specified, a log file named `Archive_Transactions_matchType_JOBID.log` is automatically generated.
- `filename`, optionally, is the name of a .ZIP file that should contain the archived transactions. If not specified, the command, by default, creates `Archived_Transactions_matchType_JOBID.zip`. Use the downloadFile command to download this file to a local computer.

> **Note:**
>
> This command runs the Archive TM Transaction job using the parameters you specify. The job ID is returned in the command output to facilitate its use with the purgeArchivedTmTransactions command. You can monitor the job from the Job Console.

Examples

- Archive old matched transactions without using filters, but using custom log and .ZIP file name:
  ```
  epmautomate archiveTmTransactions cashrecon 180 logFile=tmlogs.log
  filename=trans.zip
  ```
- Archive old matched transactions using filters:
  - `epmautomate archiveTmTransactions cashrecon 180 filterOperator=equals filterValue=101-120 FilterValue=102-202`
  - `epmautomate archiveTmTransactions cashrecon 180 filterOperator=contains filterValue=11`

assignRole

Assigns a role to users (including the user who runs this command) whose login IDs are included in an ANSI or UTF-8 encoded CSV file. Use this command to assign users to a predefined role or to an application role.

Before using this command, use the uploadFile command to upload files to an environment. The file format is as follows:

```
User Login
jane.doe@example.com
jdoe
```

See Assigning One Role to Many Users in *Getting Started with Oracle Cloud*.

> **Note:**
>
> - User Login values included in the file are not case-sensitive.
> - Use double quotation marks to enclose role names that contain space character.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

To assign predefined roles:

- Classic environments: Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

- OCI environments: Service Administrator, or Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

To assign application roles: Service Administrator or Access Control - Manage

Usage

`epmautomate assignRole FILE_NAME ROLE` where:

- *`FILE_NAME`* is the name of a CSV file containing user login IDs. Specify the CSV extension in lower case.
- `ROLE` is one of the following. This value is not case-sensitive:
  - If you are assigning users to predefined identity domain roles, `ROLE` should identify a predefined role applicable to the service. See Understanding Predefined Roles in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.
    For a description of these roles, see Managing Role Assignments at the Application Level in *Administering Access Control for Oracle Enterprise Performance Management Cloud*
  - If you are assigning users to application roles, `ROLE` should identify a role belonging to the application in the current environment. Application roles are listed in the **Roles** tab of Access Control. For a description of application roles for each business process, see these topics in *Administering Access Control for Oracle Enterprise Performance Management Cloud*:
    - Account Reconciliation
    - Enterprise Profitability and Cost Management
    - Planning, FreeForm, Financial Consolidation and Close, and Tax Reporting
    - Profitability and Cost Management
    - Oracle Enterprise Data Management
    - Narrative Reporting

Examples

- Assign users to a predefined identity domain role:
  ```
  epmautomate assignRole admin_role_file.csv "Service Administrator"
  ```
- Assign users to an application role:
  ```
  epmautomate assignRole example_file.csv "Task List Access Manager"
  ```

autoPredict

Generates predictions of future performance based on an existing Auto Predict definition in Planning or Planning Modules.

This command initiates a job that uses the historical data for each member identified in the Auto Predict definition specified in the application. For detailed information on the applications that use the Auto Predict feature, and setting up predictions, see Setting Up Predictions to Run Automatically with Auto Predict in *Administering Planning*.

Applies to

Planning, Planning Modules, if Hybrid Oracle Essbase cubes are enabled in the application.

Required Roles

Service Administrator

Usage

`epmautomate autoPredict` *`PREDICTION_DEFINITION`* `[forceRun=true|false] [paginatedDim=`*`DIMENSION_NAME`*`]` where:

- *`PREDICTION_DEFINITION`* is the name of an auto prediction definition available in the application.
- `forceRun`, optionally, specifies whether to run the prediction if the underlying definition has not changed after the initial run. Default is `false`
  Set the value of this parameter to `true` to run the Auto Predict job even if there is no change in the job definition. Use the default (`false`) to run the prediction once, at the very first time the job is executed.
- `paginatedDim`, optionally, specifies a dimension that is to be used to speed up the Auto Predict job by running predictions in parallel, in separate threads. For these parallel threads to be efficient, specify a dimension that will result in evenly spread data for each prediction thread.

Example

```
epmautomate autoPredict ASOtoBSO forceRun=true paginatedDim=Entity
```

calculateModel

Runs the calculation process in Enterprise Profitability and Cost Management applications.

Applies to

Enterprise Profitability and Cost Management

Required Roles

Service Administrator

Usage

`epmautomate calculateModel` *`POV_NAME MODEL_NAME EXECUTION_TYPE`* `[povDelimiter=`*`DELIMITER`*`] [optimizeForReporting=true|false] [captureDebugScripts=true|false] [comment=`*`COMMENT`*`] [PARAMETER=`*`VALUE`*`]`, where:

- *`POV_NAME`* is the name of the data POV to be calculated. To calculate multiple POVs, list POV names separated by a comma as the delimiter. Do not use any other delimiter to separate POV names. Enclose the list of POV names in double quotes when there are spaces in member names.
- *`MODEL_NAME`* is the name of the model to be calculated. Enclose the model name in double quotes if the name contains spaces.
- *`EXECUTION_TYPE`* is one of the following, which identifies rule execution type.
  - `ALL_RULES` to use all rules to calculate the POV.
    If you specify this value, do not specify rule subset or single rule related runtime parameters such as `rulesetSeqNumStart`, `rulesetSeqNumEnd`, and `ruleName`.
  - `RULESET_SUBSET` to use a subset of a ruleset to calculate the POV.

If you use this value, you must specify `rulesetSeqNumStart` and `rulesetSeqNumEnd` values as runtime parameters.

  - `SINGLE_RULE` to run a specific rule to calculate the POV.
    If you use this value, you must only specify a `ruleName` as the runtime parameter.
  - `RUN_FROM_RULE` to run calculations on a POV starting from a specific rule.
    If you use this value, you must only specify a `ruleName` as the runtime parameter.
  - `STOP_AFTER_RULE` to stop calculating the POV after a specific rule has finished calculations.
    If you use this value, you must only specify a `ruleName` as the runtime parameter.

- `povDelimiter`, optionally, is the delimiter used in POV values. The default delimiter is _ (under score). Delimiter must be enclosed in double quotation marks. Only these delimiters are supported:
  - _ (under score)
  - # (hash)
  - & (ampersand)
  - ~ (tilde)
  - % (percentage)
  - ; (semicolon)
  - : (colon)
  - - (dash)
- `optimizeForReporting=true|false`, optionally, specifies whether calculations are to be run with or without optimization for reporting. Default is `false`.
  Set this value to `false` to save processing time by skipping the aggregation creation step; for example, when running a single rule or a sequential series of POVs. When running multiple concurrent calculation jobs, set `optimizeForReporting=true` for all jobs, so only the last job to finish performs aggregation, avoiding redundant processing and preventing running jobs from slowing down.
- `captureDebugScripts=true|false`, optionally, identifies whether to generate debug scripts in the inbox. Oracle may need these scripts to troubleshoot calculation issues. Default is `false`.
- `comment="COMMENT"`, optionally, specifies a comment about the process in double quotation marks.
- `PARAMETER=`*`VALUE`*, optionally, indicates runtime parameters and their values to run the calculation. Specify as many parameter and value pairings as the process require. Valid parameters and their values:
  - `rulesetSeqNumStart` the sequence number of the first rule in the ruleset to be run. Valid only if `EXECUTION_TYPE=RULESET_SUBSET` is used.
  - `rulesetSeqNumEnd` specifies the sequence number of the last rule in the ruleset to be run. Valid only if `EXECUTION_TYPE=RULESET_SUBSET` is used.
  - `ruleName` name of the rule to be run. Enclose the value in double quotation marks if it contains the space character. Valid only if the value of `EXECUTION_TYPE` is set to `SINGLE_RULE`, `RUN_FROM_RULE`, or `STOP_AFTER_RULE`.
  - `clearCalculatedData=true|false` specifies whether to clear existing calculations. Default is `false`.

– `executeCalculations=true|false` specifies whether to run calculations. Default is `false`.

> **Note:**
>
> Parameter values (`true` and `false`) must be in all lower case.

Examples

- Run all rules to calculate a single POV:
```
epmautomate calculateModel FY21_Jan_Actual_Working ForecastingModel ALL_RULES
clearCalculatedData=true executeCalculations=true optimizeForReporting=true
comment="Running all rules to calculate a POV"
```
- Run all rules to calculate multiple POVs :
```
epmautomate calculateModel
"FY21:Jan:Actual:Working,FY21:Feb:Actual:Working,FY21:Mar:Actual:Working" "10
Actuals Allocation Process" ALL_RULES clearCalculatedData=true
executeCalculations=true optimizeForReporting=true captureDebugScripts=true
comment="Test calculation of many POVs" povDelimiter=":"
```
- Run a subset of a ruleset to calculate the POV:
```
epmautomate calculateModel FY21_Jan_Actual_Working ForecastingModel
RULESET_SUBSET rulesetSeqNumStart=10 rulesetSeqNumEnd=20
clearCalculatedData=true executeCalculations=true comment="Running a subset of
rules to calculate a POV"
```
- Run a specific rule to calculate the POV:
```
epmautomate calculateModel FY21_Jan_Actual_Working ForecastingModel
SINGLE_RULE ruleName="Occupancy Expense Allocations" clearCalculatedData=true
executeCalculations=true comment="Running a specific rule to calculate a POV"
```
- Run all rules to calculate a single POV using a custom POV delimiter:
```
epmautomate calculateModel FY21:Jan:Actual_Working ForecastingModel ALL_RULES
clearCalculatedData=true executeCalculations=true optimizeForReporting=true
comment="Running all rules to calculate a POV" povDelimiter=":"
```
- Run all rules to calculate POVs and model with space in names:
```
epmautomate calculateModel "FY21_Jan_New
Actual_Working,FY21:Feb:Actual:Working" "Forecasting Model" ALL_RULES
clearCalculatedData=true executeCalculations=true optimizeForReporting=true
comment="Running all rules to calculate a POV"
```

clearCube

Deletes specific data from input and reporting cubes using the settings specified in a job of type `clear cube`.

This command does not delete the application definition in the application's relational tables. See Clearing Cubes in *Administering Planning*.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate clearCube JOB_NAME` where: *`JOB_NAME`* is the name of a job defined in the application.

Example

```
epmautomate clearCube ClearPlan1
```

clearDataByPointOfView

Clears data for a specific POV for an Enterprise Profitability and Cost Management cube.

Applies to

Enterprise Profitability and Cost Management

Required Roles

Service Administrator

Usage

`epmAutomate clearDataByPointOfView` *`POV_NAME`* `[cubeName=`*`CUBE_NAME`*`]` `[`*`PARAMETER=VALUE`*`]` where:

- *`POV_NAME`* is the name of a POV in the application.
- `cubeName`, optionally, is the name of the cube in which data is to be cleared. Default is `PCM_CLC`.
- `PARAMETER=`*`VALUE`* indicates optional runtime parameters and their values. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:
  - `povDelimiter` is the delimiter used in POV values. Default is `::` (double Colon). This value must be enclosed in double quotation marks. Example: `povDelimiter="_"`. Other than the default, only these delimiters are supported: _ (under score), # (hash), & (ampersand), ~ (tilde), % (percentage), ; (semicolon), : (colon), - (dash).
  - `clearInput=true|false` specifies whether to clear input data. Default is `false`.
  - `clearAllocatedValues=true|false` specifies whether to clear allocated values. Default is `false`.
  - `clearAdjustmentValues=true|false` specifies whether to clear adjustment values. Default is `false`.

> **Note:**
>
> * Parameter values (`true` or `false`) must be in all lower case.
> * At least one of `clearInput`, `clearAllocatedValues` or `clearAdjustmentValues` parameters must be set to `true`.

Examples

- Clear data from a POV in the default PCM_CLC cube using the default POV delimiter:
  ```
  epmAutomate clearDataByPointOfView FY21::Jan::Actual::Working clearInput=true
  clearAllocatedValues=true clearAdjustmentValues=true
  ```
- Clear input data and allocated values from a POV in a specific cube using a custom POV delimiter:
  ```
  epmAutomate clearDataByPointOfView FY21_Jan_Actual_Working cubeName=PCM_REP
  povDelimiter="_" clearInput=true clearAllocatedValues=true
  ```
- Clear input data from a POV in a specific cube using a custom POV delimiter:
  ```
  epmAutomate clearDataByPointOfView FY21:Jan:Actual:Working cubeName=PCM_REP
  povDelimiter=":" clearInput=true
  ```

clearDataByProfile

Clears data from the items (for example, regions) identified in a Clear Data Profile defined in Financial Consolidation and Close and Tax Reporting.

Applies to

Financial Consolidation and Close, Tax Reporting

Required Roles

Service Administrator

Usage

`epmautomate clearDataByProfile` *`PROFILE_NAME`* where *`PROFILE_NAME`* is the name of a Clear Data Profile.

Example

```
epmautomate clearDataByProfile clearDataProfile_01
```

clearPOV

Clears model artifacts and data from a Point of View (POV) combination or a data region within the POV in an Profitability and Cost Management application.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate clearPOV` *`APPLICATION_NAME POV_NAME [QUERY_NAME]`* `PARAMETER=`*`VALUE`* `stringDelimiter="`*`DELIMITER`*`"` where:

- *`APPLICATION_NAME`* is the name of an Profitability and Cost Management application
- *`POV_NAME`* is a POV in the application. This value is required.

- `QUERY_NAME`, optionally, is the name of a query exactly as defined in Profitability and Cost Management. If specified, this query will be used to clear data region within the POV.

  > **Note:**
  >
  > If you specify a query name, you must set the value of all runtime parameters (see below) to false.

- `PARAMETER=VALUE` indicates runtime parameters and their values to clear the POV. Specify as many parameter and value pairings as the process requires. Valid parameters, at least one of which is required, and their values:
  - `isManageRule=true|false` specifies whether to clear rules
  - `isInputData=true|false` specifies whether to clear input data
  - `isAllocatedValues=true|false` specifies whether to clear allocation values
  - `isAdjustmentValues=true|false` specifies whether to clear adjustment values

  > **Note:**
  >
  > Parameter values (`true` or `false`) must be in all lower case.
  >
  > To clear data regions in a POV (if a `QUERY_NAME` is specified), you must set the value of runtime parameters (`isManageRule`, `isInputData`, `isAllocatedValues`, and `isAdjustmentValues`) to `false`.

- `stringDelimiter="DELIMITER"` specifies the delimiter used in POV values. Delimiter must be enclosed in double quotation marks. Default value is _ (underscore)

Examples

- Clear all model artifacts and data from a POV: `epmautomate clearPOV BksML12 2012_Jan_Actual isManageRule=true isInputData=true isAllocatedValues=true isAdjustmentValues=true stringDelimiter="_"`
- Clear data region within a POV: `epmautomate clearPOV BksML12 2012_Jan_Actual queryName=BksML12_2012_Jan_clear_query isManageRule=false isInputData=false isAllocatedValues=false isAdjustmentValues=false stringDelimiter="_"`

cloneEnvironment

Clones the current environment and, optionally, identity domain artifacts (users and predefined role assignments), Data Management records, audit records, Job Console records, contents of the inbox and outbox, and stored snapshots. This command is an alternative to using the Clone Environment feature in Migration.

Initiate cloning after the scheduled daily maintenance of the source and target environments. If the daily maintenance of the source environment starts while cloning is in progress, the cloning process will be terminated. The cloning process on the target environment is not affected even if the cloning is in progress at the start time of the daily maintenance. In this scenario, the daily maintenance will run after the cloning is complete.

If the cloning of your environment takes a long time, reschedule the daily maintenance start time on the source environment to avoid the cloning process from being terminated. See these information sources for information on resetting the daily maintenance start time.

- setDailyMaintenanceStartTime
- Managing Daily Maintenance in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*
- Viewing and Setting the Daily Maintenance Window Time in *REST API for Oracle Enterprise Performance Management Cloud*

Note:

- **Account Reconciliation:** After cloning, the target Account Reconciliation application settings will reset to their default values. If you wish to retain the target application settings, export them from the source environment using the exportARApplicationProperties command. Then, after the cloning is complete, import the application properties into the target environment using the importARApplicationProperties command.
- **Data Management:** Cloning of Data Management records may take a long time if the staging tables contain a very large number of records. Similarly, cloning the contents of the inbox and outbox, and stored snapshots may take considerable time, especially if they contain a large amount of data.
- **Legacy Environments:** Cloning maintains the current Oracle Essbase version as discussed in these scenarios:
  - Scenario 1: You are cloning a source legacy environment that uses an Essbase version that does not support Hybrid cubes to a target legacy environment that uses an Essbase version that supports Hybrid cubes. In this scenario, the Essbase in the target environment is downgraded to match the version in the source environment.
  - Scenario 2: You are cloning a source legacy environment that uses an Essbase version that supports Hybrid cubes to a target legacy environment that uses an Essbase version that does not support Hybrid cubes. In this scenario, the Essbase in the target environment is upgraded to match the version in the source environment.
  - Scenario 3: You are cloning a source legacy environment that uses an Essbase version that does not support Hybrid cubes to a target EPM Standard Cloud Service or EPM Enterprise Cloud Service environment, which, by default, uses an Essbase version that supports Hybrid cubes. In this scenario, the Essbase in the target environment is not downgraded to match the version in the source environment.
- **Planning:** Cloning may fail if the Planning business process contains a renamed seeded period member that has been supplanted by a custom period member. For example, you renamed the seeded *YearTotal* Period member to *unused_YearTotal* and then added an alternate type period member with the original seeded member name (*YearTotal* in this example). In this scenario, cloning of the environment may fail.

For detailed information on these topics, see Cloning EPM Cloud Environments in *Administering Migration for Oracle Enterprise Performance Management Cloud*.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Identity Domain Administrator role is required to clone users and predefined roles.

Usage

`epmAutomate cloneEnvironment` *`TARGET_USERNAME TARGET_PASSWORD TARGET_URL`* `[SnapshotName=`*`NAME`*`] [UsersAndPreDefinedRoles=true|false] [DataManagement=true|false] [appAudit=true|false] [jobConsole=true|false] [storedSnapshotsAndFiles=true|false] [DailyMaintenanceStartTime=true|false],` where:

> **Note:**
>
> - The `dataManagement` parameter does not apply to Oracle Enterprise Data Management Cloud and Narrative Reporting environments.
>   Clone Data Management records only if both the source and target environments are on the same monthly update or the target environment is one update newer than the source environment. For example, you can clone 22.01 Data Management records to another 22.01 environment or to a 22.02 environment only.
> - The `jobConsole` parameter applies to Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning only.
> - The `appAudit` parameter applies to Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning only.
>   Audit information for Financial Consolidation and Close and Tax Reporting is, by default, included in the snapshot.
> - If `dataManagement`, `jobConsole`, or `appAudit` parameter is not applicable to an environment, EPM Automate ignores the value you specify.

- *`TARGET_USERNAME`* is the ID of a Service Administrator in the target environment. You must use the target identity domain user name (not the SSO user name). If you plan to clone user and role assignments in the target environment, this user must have the Identity Domain Administrator role also.
- *`TARGET_PASSWORD`* is the location of the encrypted password file of the user identified by *`TARGET_USERNAME`*.
- *`TARGET_URL`* is the `EPM-CLOUD_BASE_URL` of the environment that will become the cloned environment.

- *SnapshotName*, optionally, is the name of a snapshot that should be used for cloning. This snapshot must be present in the source environment. Default is `Artifact Snapshot`, which uses the last maintenance snapshot to clone the environment.
- *UsersAndPreDefinedRoles*, optionally, identifies whether to clone users and their predefined role assignments (Access Control groups are always cloned). Default is `false`. For this option to work, the user identified by *TARGET_USER_NAME* must have the Identity Domain Administrator role in the target environment.

  Import of users and their predefined roles will fail if a user who is not an Identity Domain Administrator clones an environment after selecting this check box. The following error is recorded in the Migration Status Report: `Failed to import External Directory Artifact <artifact_name>. User <user_name> is not authorized to perform this operation. The user needs to have Identity Domain Administrator role to perform this operation.`

  - If you are not importing users and a user in the source snapshot is not assigned to a predefined role on the target environment, an error (`EPMIE-00070: Failed to find user during assigned roles import`) is displayed.
  - The Identity Domain Administrator role assignment is not cloned. Users with only the Identity Domain Administrator role assignment are not cloned to the target environment.
    Users assigned to a combination of Identity Domain Administrator role and predefined roles in the source environment are cloned, but assigned only to the respective predefined roles in the target environment. These users will not have the Identity Domain Administrator role in the target environment.
  - Changes to the predefined roles of the user will be updated based on the roles assigned in the source snapshot. However, role assignments in the target will not be removed to match those in the source snapshot. For example, assume that `jdoe` is assigned to the Power User predefined role in the target environment, but has only the User role in the source snapshot. In this situation, this command assigns `jdoe` to the User role and does not remove the Power User role assignment in the target environment.
  - This command does not delete existing users from the target environment if they don't exist in the source snapshot. For example, `jdoe` has an account in the target environment, but this account is not present in the source snapshot. In this situation, the account of `jdoe` in the target environment is not deleted.
  - This command adds users that do not exist in the target environment; it does not update current user properties in the target environment even if those are different in the source snapshot. For example, if the last name of `jdoe` in the source snapshot is spelled differently in the target environment, the change will not be made in the target environment. A random password is assigned to new users in the target environment. New users will receive account activation emails prompting them to change passwords.
  - This command does not change existing users' passwords in the target environment even if it is different in the source snapshot.
- `dataManagement=true|false`, optionally, clones Data Management records in the source environment to the target environment. Default is `true`, which clones Data Management records. Set this value to `false` if you do not want to clone Data Management records.
- `appAudit=true|false`, optionally, clones the audit records in the source environment to the target environment. Default is `true`, which clones application audit data. Set this value to `false` if you do not want to clone application audit data to the target environment.

- `jobConsole=true|false`, optionally, clones the Job Console records in the source environment to the target environment. Default is `true`. Set this value to `false` if you do not want to clone Job Console records.
- `storedSnapshotsAndFiles`, optionally, identifies whether the command should clone the contents of inbox and outbox, and stored snapshots. Default is `false`.

> **Note:**
>
> Only the top level folders in the inbox and outbox are cloned; subfolders are not. If you need to retain the contents of the subfolders, back them up to a local computer and then upload them to the target environment.

- *`DailyMaintenanceStartTime`*, optionally, resets the maintenance start time of the cloned target environment to that of the source environment. Default is `true`. To keep the current maintenance start time of the target environment, set this value to `false`.

Examples

- Clone the environment, users and predefined role assignments, audit data, job console records, and Data Management records. Also change the maintenance start time of the target environment to that of the source environment:
```
epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com UsersAndPreDefinedRoles=true
```
- Clone the environment including the contents of inbox and outbox, stored snapshots, but not the users and predefined role assignments, Data Management records, audit data, and Job Console records, without changing the maintenance start time of the target environment:
```
epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com DataManagement=false appAudit=false jobConsole=false storedSnapshotsAndFiles=true DailyMaintenanceStartTime=false
```
- Clone the entire environment (users and predefined role assignments, audit data, Job Console records, inbox and outbox contents, stored snapshots, and Data Management records) using a custom snapshot. Also change the maintenance start time of the target environment to that of the source environment:
```
epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com UsersAndPreDefinedRoles=true storedSnapshotsAndFiles=true SnapshotName=SampleSnapshot
```

copyDataByPointOfView

Copies data from a source POV in a cube to a destination POV in the same or another Enterprise Profitability and Cost Management cube.

Applies to

Enterprise Profitability and Cost Management

Required Roles

Service Administrator

Usage

`epmAutomate copyDataByPointOfView` *`SOURCE_POV_NAME TARGET_POV_NAME`* `copyType=ALL_DATA|INPUT` *`SOURCE_CUBE_NAME TARGET_CUBE_NAME`* `[`*`PARAMETER=VALUE`*`]` where:

- *`SOURCE_POV_NAME`* is the name of the source POV from which data is to be copied.
- *`TARGET_POV_NAME`* is the name of a valid target POV to which the data from the source is to be copied.
- `copyType` identifies the data to be copied from the source POV. Valid values are:
  - `ALL_DATA` to copy all input and calculated data to the destination POV.
  - `INPUT` to copy all input data, including driver data, to the destination POV.
- *`SOURCE_CUBE_NAME`* is the name of the cube that contains the source POV.
- *`TARGET_CUBE_NAME`* is the name of the cube that contains the target POV.
- `PARAMETER=`*`VALUE`* indicates optional runtime parameters and their values. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:
  - `povDelimiter`, optionally, is the delimiter used in POV values. Default is `::` (double Colon). This value must be enclosed in double quotation marks. Example: `povDelimiter="_"`.
    Other than the default, only these delimiters are supported: _ (under score), # (hash), & (ampersand), ~ (tilde), % (percentage), ; (semicolon), : (colon), - (dash).
  - `createDestPOV=true|false` specifies whether to create the target POV if it does not exist. Default is `false`. You must set this parameter value to `true` if the destination POV does not exist.

Examples

- Copy all data to a different POV in the same cube:
  ```
  epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
  FY22_Jan_Actual_Working ALL_DATA PCM_CLC PCM_CLC povDelimiter="_"
  createDestPOV=true
  ```
- Copy all data to a different POV in a different cube:
  ```
  epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
  FY22_Jan_Actual_Working ALL_DATA PCM_CLC PCM_REP povDelimiter="_"
  createDestPOV=true
  ```
- Copy input data to a different POV in the same cube:
  ```
  epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
  FY22_Jan_Actual_Working INPUT PCM_CLC PCM_CLC povDelimiter="_"
  createDestPOV=true
  ```
- Copy input data to a different POV in a different cube:
  ```
  epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
  FY22_Jan_Actual_Working INPUT PCM_CLC PCM_REP povDelimiter="_"
  createDestPOV=true
  ```

copyDataByProfile

Copies data for the items (for example, regions) identified in a Copy Data Profile.

Applies to

Financial Consolidation and Close, Tax Reporting

Required Roles

Service Administrator

Usage

`epmautomate copyDataByProfile` *`PROFILE_NAME`* where *`PROFILE_NAME`* is the name of a Copy Data Profile defined in Financial Consolidation and Close and Tax Reporting.

Example

```
epmautomate copyDataByProfile copyDataProfile_01
```

copyFileFromInstance

Copies a file from a source environment to the environment from which you are executing this command.

Before running this command, using EPM Automate, sign into the environment into which you want to copy the file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate copyFileFromInstance` *`SOURCE_FILE_NAME USERNAME PASSWORD_FILE URL TARGET_FILE_NAME`* where:

- *`SOURCE_FILE__NAME`* is the name of the file (including extension) that you want to copy from the source environment.
- *`USERNAME`* is the user name of a Service Administrator in the of the source environment.
- *`PASSWORD_FILE`* is the name and location of the file containing the encrypted password of the Service Administrator of the source environment.
- *`URL`* is the URL of the source environment.
- *`TARGET_FILE_NAME`* is a unique name for the file (including extension) in the environment from which you run this command.

Example

```
epmautomate copyFileFromInstance "my data file.zip" serviceAdmin
C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com "my
target data file.zip"
```

copyFromObjectStorage

Copies a file or backup snapshot from an Oracle Object Storage bucket to the current environment.

If you are copying a backup snapshot, this command copies it from the Object Storage bucket and extracts its contents in Oracle Enterprise Performance Management Cloud.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate copyFromObjectStorage` *`USERNAME PASSWORD URL TARGET_FILE_NAME`* where:

- *`USERNAME`* is the ID of a user who has the required access rights in Oracle Object Storage Cloud.
  For users created in a federated identity provider, specify the fully-qualified name of the user (for example, `exampleIdP/jdoe` or `exampleIdP/john.doe@example.com`, where `exampleIdP` is the name of the federated identity provider). For other users, provide the User ID.
- *`PASSWORD`* is the Swift password or auth token associated with the user. This password is not the same as the password that you use to sign into the Object Storage Console. Auth token is an Oracle-generated token that you use to authenticate with third-party APIs, for example to authenticate with a Swift client. For instructions to create this token, see To create an auth token in *Oracle Cloud Infrastructure Documentation* .
- *`URL`* is the URL of the Oracle Object Storage Cloud bucket, including the bucket name and the name of the object to be copied.
  The URL format:

  `https://`*`swiftobjectstorage.region_identifier`*`.oraclecloud.com/v1/`*`namespace/bucket_name/object_name`*

  Components of this URL:

  - `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, `axaxnpcrorw5`, 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 name of the snapshot or file that you want to copy from Oracle Object Storage Cloud. This value must exactly match the full name of the object in Object Storage Cloud. Do not use an extension such as `.zip` unless the object name contains it.

For more information, see these topics in *Oracle Cloud Infrastructure Documentation*

- Regions and Availability Domains
- Understanding Object Storage Namespaces
- Managing Buckets

- *`TARGET_FILE_NAME`* is a unique name for the file or snapshot in the EPM Cloud environment. When copying snapshots, do not specify the ZIP extension so that this file name can be used with the importSnapshot command.
  Files larger than 100 MB are stored in Oracle Object Storage within a logical directory along with the manifest file that identifies its segments. Specify the name of the logical directory as the *`TARGET_FILE_NAME`*.

Examples

In these examples, replace *`URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET`* with a working URL in this format: `https://swiftobjectstorage.`*`region_identifier`*`.oraclecloud.com/v1/`*`namespace/bucket_name/`*.

- Copy a snapshot named `backup_Snapshot_12_05_20.zip` from Oracle Object Storage bucket to EPM Cloud and rename it:
  ```
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/backup_Snapshot_12_05_20.zip snapshot_from_osc
  ```
- Copy a snapshot named `backup_Snapshot_12_05_20` from Oracle Object Storage bucket to EPM Cloud and rename it:
  ```
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/backup_Snapshot_12_05_20 snapshot_from_osc
  ```
- Copy a snapshot named `backup_Snapshot_12_05_20` from Oracle Object Storage bucket to EPM Cloud without renaming it:
  ```
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/backup_snapshot_12_05_20 backup_snapshot_12_05_20
  ```
- Copy a file to EPM Cloud from Oracle Object Storage bucket:
  ```
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/example_file.txt copied_from_osc.txt
  ```

copyOwnershipDataToNextYear

Copies the ownership data from the last period of a year to the first period of the next year.

Initial default and override ownership settings are automatically carried forward from period to period within the same year but not to the periods in subsequent years. To carry the most current ownership settings from the last period in a year to the first period of the next year, you must copy the ownership settings from the last period of the year in the POV, to the first period of the next year.

Applies to

Financial Consolidation and Close and Tax Reporting.

Required Roles

Service Administrator, Power User, User

Usage

`epmautomate copyOwnershipDataToNextYear` *`Scenario Year`* where:

- `Scenario` is the name of the scenario from which ownership data is to be copied.
- `Year` is the year from which ownership data is to be copied to the first period of next year.

Example

```
epmautomate copyOwnershipDataToNextYear FCCS_total_Actual FY18
```

copyPOV

Copies the model artifacts and Oracle Essbase cube data from a source POV to a destination POV.

Applies to

Profitability and Cost Management.

Required Roles

Service Administrator, Power User

Usage

`epmautomate copyPOV` *`APPLICATION_NAME SOURCE_POV_NAME TARGET_POV_NAME`* `PARAMETER=`*`VALUE`* `stringDelimiter="`*`DELIMITER`*`" [isInputData=true|false isAllInputData=true|false]` where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that contains the source POV.
- *`SOURCE_POV_NAME`* is the name of the source POV in the specified application
- *`TARGET_POV_NAME`* is the name of a valid target POV in `Draft` status
- `PARAMETER=`*`VALUE`* indicates runtime parameters and their values to copy the POV. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:
  - `isManageRule=true|false` specifies whether to copy rules.
  - `isInputData=true | isAllData=true | isAllInputData=true` optionally specifies how to copy data. For these parameters, default value is false. Specify only one of these as true:
    * specify `isInputData=true` to copy input data to the destination POV.
    * specify `isAllData=true` to copy all input and calculated data to the destination POV.

    - `AllInputData=true` to copy all input data, including driver data, to the destination POV.
  - `modelViewName=NAME` specifies the name of the data slice that is to be copied from the source POV to the target POV.
  - `createDestPOV=true|false` specifies whether to create the target POV if it does not exist.
  - `nonEmptyTupleEnabled=true|false` specifies whether to enable Non-Empty Tuple (NET) so that the command considers only the intersections that have data. Default is `true`, which, in rare cases, may cause the command to not perform well for copying Essbase data. In those cases, override the default by using `nonEmptyTupleEnabled=false` to improve performance.

> **Note:**
>
> Parameter values (`true` or `false`) must be in all lower case.

- `stringDelimiter="DELIMITER"` specifies the delimiter used in POV values. Delimiter must be enclosed in double quotation marks.

Examples

- `epmautomate copyPOV BksML12 2012_Jan_Actual 2012_Feb_Actual isManageRule=true isInputData=true modelViewName="Balancing - 5 Customer Costs" createDestPOV=true stringDelimiter="_"`
- `epmautomate copyPOV BksML12 2012_Jan_Actual 2012_Feb_Actual isManageRule=true isAllInputData=true createDestPOV=true stringDelimiter="_"`
- `epmautomate copyPOV BksML12 2012_Jan_Actual 2012_Feb_Actual isManageRule=true isAllData=true createDestPOV=true stringDelimiter="_"`

copySnapshotFromInstance

Copies the current snapshot from a source environment to the environment (target) from which you run this command.

This command is primarily used as the first step in migrating an environment by copying the current snapshot from another environment; for example from a test environment to a production environment. You use the importSnapshot command to complete the migration process.

Before running this command, start an EPM Automate session and sign into the target environment.

If this command is executed to copy the current snapshot while the snapshot of the source environment is being generated; for example, during the daily maintenance, you will receive the `File not found` error.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate copySnapshotFromInstance SNAPSHOT_NAME USERNAME PASSWORD_FILE URL` where:

- *SNAPSHOT_NAME* is the name of an existing snapshot in the source environment.
- *USERNAME* is the user name of a Service Administrator of the source environment.
- *PASSWORD_FILE* is the name and location of the file containing the encrypted password of the Service Administrator of the source environment.
- *URL* is the URL of the source environment.

Example

```
epmautomate copySnapshotFromInstance "Artifact Snapshot" serviceAdmin
C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com
```

copyToObjectStorage

Copies a file or snapshot from the current environment to an Oracle Object Storage Cloud bucket.

If you are copying a snapshot, this command zips the content of the snapshot before copying it to Oracle Object Storage.

To facilitate fast copying of files, this command chunks large files (larger than 100 MB) into 10 MB segments (named *FILE_NAME*/*FILE_NAME*_object_store_bytes_seg_0 through *FILE_NAME*/*FILE_NAME*_object_store_bytes_seg_*n*) and creates a manifest file (named *FILE_NAME*/*FILE_NAME*.manifest). File segments are stored in Oracle Object Storage along with the manifest file. In the Object Storage Console, the file is displayed as a logical directory containing the file segments and the manifest file.

Files smaller than 100 MB are not segmented and are stored with the original file name.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate copyToObjectStorage SOURCE_FILE_NAME USERNAME PASSWORD URL` where:

- *SOURCE_FILE_NAME* is the name of the file or snapshot in Oracle Enterprise Performance Management Cloud. If you are copying a snapshot, do not specify the ZIP extension.
- *USERNAME* is the ID of a user who has the required access rights to write to Oracle Object Storage Cloud.

For users created in a federated identity provider, specify the fully-qualified name of the user (for example, `exampleIdP/jdoe` or `exampleIdP/john.doe@example.com`, where `exampleIdP` is the name of the federated identity provider). For other users, provide the User ID.

- *`PASSWORD`* is the Swift password or auth token associated with the user. This password is not the same as the password that you use to sign into the Object Storage Console. Auth token is an Oracle-generated token that you use to authenticate with third-party APIs, for example to authenticate with a Swift client. For instructions to create this token, see To create an auth token in *Oracle Cloud Infrastructure Documentation* .
- *`URL`* is the URL of the Oracle Object Storage Cloud bucket with an optional object name appended.
  The URL format without object name:

  `https://swiftobjectstorage.`*`region_identifier`*`.oraclecloud.com/v1/`*`namespace/bucket_name`*

  The URL format with object name:

  `https://`*`swiftobjectstorage.region_identifier`*`.oraclecloud.com/v1/`*`namespace/bucket_name/object_name`*

  Components of this URL:

  - `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, `axaxnpcrorw5`, 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`, optionally, is name that you want to use for the file on Oracle Oracle Object Storage Cloud. If an object name is not specified, the file will be copied with its original name.

  For more information, see these topics in *Oracle Cloud Infrastructure Documentation*

  - Regions and Availability Domains
  - Understanding Object Storage Namespaces
  - Managing Buckets

Examples

In these examples, replace *`URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET`* with a working URL in this format: `https://swiftobjectstorage.`*`region_identifier`*`.oraclecloud.com/v1/`*`namespace/bucket_name/`*.

- Copy a snapshot to an Object Storage bucket and rename it:
  ```
  epmautomate copyToObjectStorage "Artifact Snapshot"
  oracleidentitycloudservice/jDoe example_pwd
  URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/Snapshot_04_30_21
  ```
- Copy a file to an Object Storage bucket:
  ```
  epmautomate copyToObjectStorage example_file.txt oracleidentitycloudservice/
  jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET
  ```

- Copy a file to an Object Storage bucket and rename it:
  `epmautomate copyToObjectStorage example_file.txt eoracleidentitycloudservice/jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/epm_text_file.txt`

createGroups

Adds groups to Access Control using an ANSI or UTF-8 encoded CSV file that was uploaded to the environment.

You use the uploadFile command to upload files to an environment. The file format is as follows:

```
Group Name,Description
Example_grp1,My test group
Example_grp2,My other test group
```

Group names are not case-sensitive. When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate createGroups FILE_NAME` where *FILE_NAME* is the name of a CSV file containing group names and descriptions.

Example

```
epmautomate createGroups group_file.CSV
```

createNRSnapshot

Create an on-demand snapshot, named `EPRCS_Backup.tar.gz`, of a Narrative Reporting environment.

You can download `EPRCS_Backup.tar.gz` and error file to a local computer using the downloadFile command or copy it to another environment using the copyFileFromInstance command.

The application data in `EPRCS_Backup.tar.gz` is as of the last daily maintenance. If you need to backup more recent data, use the data export Narrative Reporting feature.

Applies to

Narrative Reporting

Required Roles

Service Administrator

Usage

`epmautomate createNRSnapshot [errorFile=Error_File.txt]` where `errorFile`, optionally, identifies the name of a unique text file for recording errors, if any encountered by the command.

Example

```
epmautomate createNRSnapshot errorFile=EPRCS_backup_Error.txt
```

createReconciliations

Copies the profiles to a specified period.

Applies to

Account Reconciliation.

Required Roles

Service Administrator, Power User, User, Viewer

Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate createreconciliations` *`PERIOD SAVED_FILTER`* where:

- *`PERIOD`* is the name of a period
- *`SAVED_FILTER`* is the name of a saved public filter. If you do not specify a saved filter, EPM Automate copies all applicable profiles

Examples

- Copy all profiles for the period: `epmautomate createReconciliations "January 2015"`
- Copy profiles of a specific filter: `epmautomate createReconciliations "January 2015" "Corporate Recs"`

deleteFile

Deletes a file or a snapshot from the default upload location, the inbox or outbox, a Data Management folder, or from profitinbox/profitoutbox.

To delete a file from a location other than the default upload location, you must specify the file location.

If this command is executed to delete a snapshot that is in the process of being generated or archived, you will receive one of these errors:

- `File not found` if the snapshot is being generated

- `Archieve process is in progress. Unable to Rename or Delete` if the snapshot is being archived

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate deleteFile` *`FILE_NAME`*

> **Note:**
>
> You must specify the file name including extension; for example, `data.csv`, `data.zip`, if applicable. You can delete a snapshot without specifying its file extension (.ZIP). This usage, however, is deprecated. You should specify the location of the file if it is not in the default location. For detailed information, see Default File Locations. Supported locations include `inbox`, `profitinbox`, `outbox`, `profitoutbox`, `to_be_imported`, and `inbox/`*`directory_name`*.

Examples

- Delete a file from the default upload location:
  `epmautomate deleteFile data.csv`
- Delete a file from the inbox:
  `epmautomate deleteFile inbox/data.csv`
- Delete from the outbox:
  `epmautomate deleteFile outbox/data.csv`
- Delete a snapshot that you created using Migration:
  - `epmautomate deleteFile "Backup 18-06-12.zip"` or
  - `epmautomate deleteFile "Backup 18-06-12"` (deprecated)
- Delete from profitinbox (Profitability and Cost Management):
  `epmautomate deleteFile profitinbox/data.csv`
- Delete from profitoutbox (Profitability and Cost Management):
  `epmautomate deleteFile profitoutbox/data.csv`
- Delete from a Data Management upload folder:
  `epmautomate deleteFile inbox/dm_data/data.csv`
- Delete from a Data Management folder:
  `epmautomate deleteFile outbox/dm_data/data.csv`

deleteGroups

Removes groups from Access Control based on the information available in an ANSI or UTF-8 encoded CSV file that was uploaded to the environment.

You use the uploadFile command to upload files to an environment. The file format is as follows:

```
Group Name
Example_grp1
Example_grp2
```

Group Name values in the file are not case-sensitive. When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate deleteGroups` *`FILE_NAME`* where *`FILE_NAME`* is the name of a CSV file containing the names of the groups to be removed from Access Control.

Example

```
epmautomate deleteGroups group_file.CSV
```

deletePointOfView

Deletes artifacts and Oracle Essbase cube data from a POV in an Enterprise Profitability and Cost Management application.

Applies to

Enterprise Profitability and Cost Management

Required Roles

Service Administrator

Usage

`epmautomate deletePointOfView` *`POV_NAME`* `[povDelimiter="`*`DELIMITER`*`"]` where:

- *`POV_NAME`* identifies the name of the POV to be deleted.
- `povDelimiter` is the delimiter used in POV values. Default is `::` (double Colon). This value must be enclosed in double quotation marks. Example: `povDelimiter="_"`.

Other than the default, only these delimiters are supported: _ (under score), # (hash), & (ampersand), ~ (tilde), % (percentage), ; (semicolon), : (colon), - (dash).

Example

- Deleting a POV that uses a custom POV delimiter
  `epmAutomate deletePointOfView FY21_Jan_Actual_Working povDelimiter="_"`
- Deleting a POV that uses the default POV delimiter
  `epmAutomate deletePointOfView FY21::Jan::Actual::Working`

deletePOV

Deletes model artifacts and Oracle Essbase cube data from a POV in Profitability and Cost Management.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate deletePOV` *`APPLICATION_NAME POV_NAME`* `stringDelimiter="`*`DELIMITER`*`"`, where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that contains the POV to be deleted.
- *`POV_NAME`* is the name of the POV to be deleted. This value is required.
- `stringDelimiter="`*`DELIMITER`*`"` specifies the delimiter used in POV values. Delimiter must be enclosed in double quotation marks.

Example

```
epmautomate deletePOV BksML12 2012_Jan_Actual stringDelimiter="_"
```

deployCube

Deploys or redeploys the calculation cube of an Profitability and Cost Management application.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate deployCube` *`APPLICATION_NAME`* `PARAMETER=`*`VALUE`* `comment="comment"` where:

- *`APPLICATION_NAME`* is the name of an Profitability and Cost Management application

- `PARAMETER=VALUE` indicates runtime parameters and their values to deploy the cube. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:

  > **Note:**
  >
  > Parameter values (`true` or `false`) must be in all lower case.

  - `isKeepData=true|false`

    specifies whether to preserve existing data, if any
  - `isReplaceCube=true|false` specifies whether to replace the existing cube

    > **Note:**
    >
    > Values of `isKeepData` and `isReplaceCube` cannot both be set to `true`.

  - `isRunNow=true|false` specifies whether to run the process right away
- `comment` is an optional comment enclosed in double quotation marks

Example

```
epmautomate deployCube BksML12 isKeepData=true isReplaceCube=false isRunNow=true
comment="Test cube deployment"
```

deployEJTemplates

Deploys finalized Enterprise Journal templates to open periods in Financial Consolidation and Close. Deploying Enterprise Journal templates creates recurring journals associated with the template for the selected period. It also allows you to create ad hoc journals using the deployed template(s).

This command is an alternative to using Financial Consolidation and Close screens to deploy new Enterprise Journal templates at the beginning of the month.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator, Power User

Usage

`epmautomate deployEJTemplates YEAR PERIOD [Template=TEPMPLATE_NAME] [ResetJournals=true|false]` where:

- `Year` is the journal year.
- `Period` is the journal period. This value can be specified only if the year is specified.

- `Template=TEMPLATE_NAME` identifies the names of the journals to be deployed. To deploy more than one journal, provide each unique template name in `Template=TEMPLATE_NAME` format, for example, `Template="Loan Details" Template="Housing Details" Template="Repayment Details"`.
  If this parameter value is not specified, the command deploys all the templates for the specified year and period combination.
- `ResetJournals` optionally indicates whether all journals must be reset to the first stage after redeploying the templates. Default is `false`.
  Financial Consolidation and Close validates this value internally based on changes to templates, and may override the value you specify, if required.

Example

```
epmautomate deployEJTemplates 2021 May Template="Loan Details" Template="Housing Details" ResetJournals=true
```

deployFormTemplates

Deploys finalized form templates to new data collection periods to create Supplemental Data Forms, ensuring a consistent and repeatable data collection process.

Applies to

Financial Consolidation and Close, Tax Reporting.

Required Roles

Service Administrator, Power User

Usage

`epmautomate deployFormTemplates COLLECTION_INTERVAL [DIMENSION] [Template] [resetWorkFlows=true|false]` where:

- `COLLECTION_INTERVAL` is the name of the collection interval to which the template is to be deployed.
- `DIMENSION`, optionally, specifies the frequency dimensions of the data collection process in `DIMENSION=MEMBER_NAME` format. Specify as many dimensions as defined in the collection interval (a maximum of four including Year and Period; for example, `"Year=2020" "Period=July" "Product=Oracle EPM" "Consolidation=entity Input"`. No default value is used if this parameter value is not specified.
- `Template`, optionally, identifies unique names for the form templates to deploy in `Template=TEMPLATE_NAME` format. You can specify any number of unique names (as many as needed) in this format. For example, `Template="Loan Details Template" Template="Housing Details Template" Template="Repayment Detals Template"`.
  If this property value is not specified, the command deploys all the templates for the specified interval.
- `resetWorkFlows`, optionally, indicates whether all forms are to be reset to the first stage after redeploying them. Default is `false`.

Example

```
epmautomate deployFormTemplates "Journal Collection Interval" "Year=2020" "Period=July" "Product=Oracle EPM" "Consolidation=entity Input" Template="Loan Details Template" Template="Housing Details Template" resetWorkFlows=true
```

deployTaskManagerTemplate

Deploys tasks from a Task Manager template into a task schedule ensuring consistent execution of repetitive business processes.

Applies to

Financial Consolidation and Close, Tax Reporting

Required Roles

Service Administrator

Usage

`epmAutomate deployTaskManagerTemplate` *`TEMPLATE_NAME SCHEDULE_NAME YEAR PERIOD DAY_ZERO_DATE`* `[dateFormat=`*`DATE_FORMAT`*`] [orgUnit=`*`ORGANIZATION UNIT`*`]` where:

- *`TEMPLATE_NAME`* is the name of Task Manager template to deploy.
- *`SCHEDULE_NAME`* is the schedule name to create from the template.
- *`YEAR`* is the Year dimension member where the template is to be deployed.
- *`PERIOD`* is the Period dimension member where the template is to be deployed.
- *`DAY_ZERO_DATE`*is the day zero date, in valid format, to be used to create the schedule.
- `dateFormat`, optionally, is the date format for day zero date. Default format is YYYY-MM-DD.
- `orgUnit`, optionally, is the name of the organization unit. If a value is not specified, the schedule will be created using the standard date mapping. Holiday rules will not be used.

Example

- Deploy Task Manager Template for *Ind* organization unit using the default date format (YYYY-MM-DD) for zero date:
  ```
  epmautomate deployTaskManagerTemplate "Vision Monthly Close" "Qtr 2 Close"
  2021 July 2021-07-10 orgUnit=Ind
  ```
- Deploy Task Manager Template for Ind organization unit using dd/mm/yyyy as the date format for zero date:
  ```
  epmautomate deployTaskManagerTemplate "Vision Monthly Close" "Qtr 2 Close"
  2021 July 02/07/2021 dateFormat=dd/MM/yyyy orgUnit=Ind
  ```

dismissIPMInsights

Automates the dismissing of Intelligent Performance Management (IPM) Insight data before running new IPM Insight jobs. Dismissing data closes all open insights on which you do not plan to take action. This command is an alternative to manually dismissing the data using the IPM Insight dashboard.

Applies to

Planning, Planning Modules, Strategic Workforce Planning, Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate dismissIPMInsights [comment="comment"]` where comment, optionally, is a justification for dismissing open insights.

Example

```
epmautomate dismissIPMInsights comment="dismissing unusable insights"
```

downloadFile

Downloads a file from an environment to the local computer.

Use this command is to download data, metadata, and back up snapshots for local storage. The file is downloaded into the folder from which you run EPM Automate.

If this command is executed to download the current snapshot while the snapshot of the environment is being generated; for example, during the daily maintenance, you will receive the `File not found` error.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

```
epmautomate downloadFile "[FILE_PATH]/FILE_NAME"
```

Examples

- Download maintenance snapshot: `epmautomate downloadFile "Artifact Snapshot"`
- Download a custom snapshot: `epmautomate downloadFile "mySnapshot.zip"`
- Download a Narrative Reporting maintenance snapshot: `epmautomate downloadFile "EPRCS_Backup.tar.gz"`
- Download a file from default download location: `epmautomate downloadFile data.csv`
- Download from a Data Management folder: `epmautomate downloadfile outbox/dm_data/data.csv`
- Download from profitoutbox: `epmautomate downloadFile profitOutbox/data.csv`

enableApp

Enables an application.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate enableapp APPLICATION_NAME` where *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that you want to enable.

Example

```
epmautomate enableApp BksML12
```

enableQueryTracking

Enables query tracking on ASO cubes to start capturing user data retrieval patterns (queries).

You use the captured data retrieval patterns to optimize ASO cube aggregation, which is initiated using the executeAggregationProcess command.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate enableQueryTracking ASO_CUBE_NAME`, where *`ASO_CUBE_NAME`* is the name of the ASO cube in which query tracking is to be activated.

Example

```
epmautomate enableQueryTracking VISION_ASO
```

encrypt

Uses Advanced Encryption Standard (`AES/CBC/PKCS5Padding(128)`) to encrypt Oracle Enterprise Performance Management Cloud password (or the OAuth2.0 refresh token and client ID for accessing OCI (Gen 2) environments), and optionally, the internet proxy server password used for signing in to Oracle Fusion Cloud EPM environments, and stores it in a password file.

Encrypting the secrets allows Service Administrators to share their encrypted password file with developers who write EPM Automate scripts so that they can execute the scripts. This precludes the need to share the Service Administrator password or create a generic, shared EPM Cloud account specifically for running scripts.

Encrypting password is a one-time process.

> **Note:**
>
> See Handling Special Characters for information on encrypting passwords that contain special characters.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate encrypt` *`PASSWORD|REFRESH_TOKEN KEY PASSWORD_FILE`* `[ClientID=`*`CLIENT_ID`*`] [ProxyServerPassword=`*`PROXY_PASSWORD`*`]` where:

- *`PASSWORD|REFRESH_TOKEN PASSWORD`* is the password or the OAuth refresh token that you want to encrypt. You cannot use corporate credentials with EPM Automate.
- *`KEY`* is the private key that is to be used to encrypt the password.
- *`PASSWORD_FILE`* is the name and location of the file that stores the encrypted password or refresh token. The password file must use the `.epw` extension.
- `ClientID`, optionally, is the client identifier that is created during OAuth 2.0 setup. This value must be specified while encrypting an OAuth 2.0 refresh token. Do not specify this value while encrypting a password.
- `ProxyServerPassword` is the password to authenticate the user with the HTTP proxy server. Required only if authentication at proxy server is enabled for your network.

Examples

- **Encrypt only EPM Cloud password:** `epmautomate encrypt P@ssword1 myKey C:\mySecuredir\password.epw`
- **Encrypt EPM Cloud and internet proxy server passwords:** `epmautomate encrypt E@xample1 myKey C:\mySecuredir\password.epw ProxyServerPassword=Proxy_Pwd1`
- **Encrypt refresh token and Client Id:** `epmautomate encrypt AAyyilYBAWD4....FVkxefd8kjoJr6HJPA= myEncyprtion42Key C:\mySecuredir\oauthfile1.epw ClientID=6fdf2e72fd343430ABR22394C`

essbaseBlockAnalysisReport

Creates the Essbase Block Analysis report that helps you analyze Oracle Essbase data to support the tuning of Block Storage Option (BSO) cubes (generally, used for calculations) in your application.

The Essbase Block Analysis report is helpful to resolve performance issues resulting from patterns of data, for example, repeated numbers in Essbase BSO cubes. The report provides information on these three areas:

1. **Percentage of blocks with only Zero**, which shows the blocks that contain only zeros as a percentage of all the blocks contained in the export file
2. **Top 10 Repeated Numerical Cell Values By Percentage of Numerical Cells** This table shows the top 10 repeated values as a percentage of all the values in the export file.
3. **Top 100 Dense Member Combinations with Repeated Values** This table shows the top 100 dense combinations with repeated values in the cube. The **Cell Value** column shows a value for each member, in the order it appears in the hierarchy, as a different column. For example, if Period is across the column, there will be a different column for January, February, and so on. Other dense dimension(s) appear in the rows. This should help you identify the locations of the repeated values.

Before running this report, use the exportEssbaseData command to export the data from the cube for which you want to create the Block Analysis report to a zip file. You may export level0 or all data as needed. Run this command to create the Block Analysis report for this zip file. The report is created in the outbox; you can use the downloadFile command to download it to a local computer or the sendMail command to email it.

Applies to

Financial Consolidation and Close, Planning, Planning Modules, FreeForm, Strategic Workforce Planning, Sales Planning, and Tax Reporting.

Required Roles

Service Administrator

Usage

`epmautomate essbaseBlockAnalysisReport` *`EXPORT_DATA_FILE`*`.zip` *`REPORT_FILE`*, where

- `EXPORT_DATA_FILE.zip` is the name of the zip file that contains the Essbase data that was previously exported from of a BSO cube using the exportEssbaseData command.
- `REPORT_FILE` is the name for the HTML formatted Block Analysis report file.

Example

```
epmautomate essbaseBlockAnalysisReport Plan1_cube_data.zip block_report.html
```

executeAggregationProcess

Initiates the aggregation process, optionally using query tracking statistics, to improve the performance of ASO cubes. This is an important step in optimizing ASO cubes.

Before running this command:

- Use the enableQueryTracking command to capture data retrieval statistics to optimize ASO aggregation.
- Allow sufficient time for the business process to capture user data retrieval patterns (queries) that can be used to create aggregate views.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate executeAggregationProcess` *`ASO_CUBE_NAME`* `[useQueryData=true|false] [includeAlternateRollups=disable|enable] [growthSizeRatio=`*`VALUE`*`]`, where:

- `useQueryData` to use recorded query data, collected using query tracking, to select the most appropriate set of aggregate views. Default is `false`.
- `includeAlternateRollups` to include secondary hierarchies (with default level usage) in the view selection process. Default is `disable`.
- `growthSizeRatio`, optionally, is the ratio for maximum cube growth to aggregate the views the server selects. The cube growth will be stopped when the maximum growth reaches the ratio that you specify. Default setting allows the cube to grow without any growth ratio limit.

> **Note:**
>
> To create default aggregate views, run this command without specifying optional parameters.

Examples

- Create aggregate view based on query data captured using the enableQueryTracking command:
  ```
  epmautomate executeAggregationProcess VISION_ASO useQueryData=true
  includeAlternateRollups=enable
  ```
- Create default aggregate view:
  ```
  epmautomate executeAggregationProcess Vis1ASO
  ```

executeBurstDefinition

Executes a bursting definition that specifies the artifacts, POVs, and other settings required to run reports or books for more than one member of a single dimension for one data source.

Applies to

Narrative Reporting

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer must be assigned additional security through ACL

Usage

`epmAutomate executeBurstDefinition` *`ARTIFACT_NAME`* where *`ARTIFACT_NAME`* is the bursting definition path and name.

Example

```
epmAutomate executeBurstDefinition "library/Reports/Example BurstDef1"
```

executeReportBurstingDefinition

Using a bursting definition, executes bursting for a single report or book for more than one member of a single dimension, and publishes a PDF or static (not refreshable in Oracle Smart View for Office) Excel output for each member.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmAutomate executeReportBurstingDefinition` *`BURST_DEFINITION_NAME`* `[jobName=`*`JOB_NAME`*`]` where:

- *`BURST_DEFINITION_NAME`* is the path and name of a bursting definition.
- *`JOB_NAME`*, optionally, is the name of the job that should be used for executing the bursting definition. Default is `Execute Bursting Definition`.

Example

```
epmAutomate executeReportBurstingDefinition /Library/MonthlySalesBurstDef
```

exportAccessControl

Exports user details report, which 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 information on their access (such as assignment to Groups, Teams, and Organizations), to a CSV or XLS file.

A sample report:

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Name**	**User Login**	**Status**	**Teams**	**Email**	**Role**	**Workflow Roles**	**Preparer**	**Reviewer**	**Organizations**	**Power User Filter**	**Last Login**
2	John Doe	john.doe@example.com	Available	AP Preparers AR Preparers	john.doe@example.com	Administrator		Yes	Yes			
3	Jane Doe	jane.doe@example.com	Available	AP Reviewers AR Reviewers	jane.doe@example.com	Power User		No	No			
4	ats_power_user4	ats_power_user4	Available		example1@example.com	User		No	No			
5	ats_user1	ats_user1	Available		example2@example.com	User		No	No			
6	ats_ view_user1	ats_view_user1	Available		view.user@exaple.com	Viewer		No	No			

You can download this report using the downloadFile command.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmAutomate exportAccessControl REPORT_NAME [reportFormat=XLS|CSV]` where:

- *`REPORT_NAME`* is the name of the export file that will contain the report.
- `reportFormat`, optionally, is the file format. Valid values are XLS and CSV (default).

Example

```
epmAutomate exportAccessControl aclreport.xls reportFormat=XLS
```

exportAppAudit

Exports data audit records into a ZIP file, which you can download and archive on a local computer. Audit information for up to 365 days is available in the environment.

The first character in the output CSV file is the Byte Order Mark (BOM) character `\ufeff` followed by an encrypted application identifier enclosed in double quotes. CSV file header follows the application identifier.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportAppAudit EXPORT_FILE_NAME [userNames=USER_NAMES] [nDays=Number_of_Days] [excludeApplicationId=true|false]`, where:

- *`EXPORT_FILE_NAME`* is the name of the ZIP file that will store the exported audit data. You use the downloadFile command to download files from an environment.
- `userNames`, optionally, is a comma separated list of user login names. If specified, only the audit data created by these users will be exported. Do not specify this value if you want to export the audit data for all users.
- `nDays`, optionally, identifies the number of days for which audit records are to be exported. Default is seven days. Possible values are: `all` to export available audit data for the last 365 days, `1`, `2`, `7`, `30`, `60`, and `180`.
- `excludeApplicationId`, optionally identifies whether the application identifier is to be written to the export file. Default is `false`.

> **Note:**
>
> Data from exported files that do not contain the application identifier cannot be imported into Oracle Enterprise Performance Management Cloud environments.

Examples

- Export audit data with the application identifier:
  ```
  epmautomate exportAppAudit auditData userNames=johnDoe,jane.doe@example.com
  ndays=30
  ```
- Export audit data without the application identifier:
  ```
  epmautomate exportAppAudit auditData userNames=johnDoe,jane.doe@example.com
  ndays=30 excludeApplicationId=true
  ```

exportAppSecurity

Exports the artifact-level access assignments (ACLs) to a CSV file, which you can download for local storage.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportAppSecurity` *`EXPORT_FILE_NAME`*`.CSV` where *`EXPORT_FILE_NAME`* is the name of the file that will store the exported security data. This file will be created in the outbox, from where you can download it to your computer.

Example

```
epmautomate exportAppSecurity app_security.CSV
```

exportARApplicationProperties

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.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmautomate exportARApplicationProperties` *`FILE_NAME`*
`[Properties=`*`PROPERTIES_TO_EXPORT`*`]`

- *FILE_NAME* is the name for the JSON file that will store the exported property values. You can download the export file using the downloadFile command. Use the uploadFile command to upload it to the target environment and then run the importARApplicationProperties command to restore these settings in the target environment.
- Properties, optionally, is a comma separated list of properties to export. You can export some or all of the following properties. If this property is omitted, all these properties are exported:
  - Theme: exports the display theme used in the environment.
  - EmailNotification: exports the email notification settings defined in the environment.
  - DisplayBusinessProcessName: 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 log image used in the environment.

Example

Export only email notification and Redwood Experience settings, and logo image from an environment:

```
epmautomate exportARApplicationProperties myProp.JSON
Properties=EmailNotification,RedwoodExperience,LogoImage
```

exportBackgroundImage

Exports the background image used in an Account Reconciliation environment to a JPG file so that you can import it into another environment.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

epmautomate exportBackgroundImage *IMAGE_NAME*.jpg, where *IMAGE_NAME* is the name for the background image file.
You can download the image file using the downloadFile command. Use the uploadFile command to upload it to the target environment and then run the importBackgroundImage command to import it.

Example

```
epmautomate exportBackgroundImage corpImage.jpg
```

exportCellLevelSecurity

Exports cell-level security settings from the business process into a ZIP file, which you can download to a local computer using the downloadFile command.

Applies to

Planning, Planning Modules, FreeForm, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportCellLevelSecurity FILE_NAME.ZIP [names=SECURITY_RECORD_NAMES]` where:

- `FILE_NAME` is the name for the ZIP file that will be created to hold the Excel file containing cell-level security information.
- `names`, optionally, identifies a comma separated list of cell-level security definitions in the application. If this option is not provided, all cell-level security definitions in the application are exported.

Examples

- **Export specific cell-level security definitions**
  ```
  epmautomate exportCellLevelSecurity ExportCLSDRecordsFile.zip
  names=CLSDAccountPeriod,CLSDEntityPeriod,CLSDProductPeriod
  ```
- **Export all cell-level security definitions**
  ```
  epmautomate exportCellLevelSecurity ExportCLSDRecordsFile.zip
  ```

exportConsolidationJournals

Exports Consolidation Journals using a job defined in Financial Consolidation and Close.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator

Usage

`epmautomate exportConsolidationJournals jobName [fileName=FILE_NAME]` where

- `jobName` is the name of a Journal Export job created in Financial Consolidation and Close.
- `fileName`, optionally, is the name of a .JLF file into which the journal entries are to be exported. Use the downloadFile command to download this file to a local computer.

Example

```
epmautomate exportConsolidationJournals "JEXPORT1" fileName=Export_Test.jlf
```

exportData

Exports application data into a ZIP file using the export data settings, including file name, specified in a job of type `export data`.

The exported data file is stored in the default download location from where you can download it to your computer. Use the Inbox/Outbox Explorer to view details of the exported file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportData JOB_NAME [FILE_NAME]` where *`JOB_NAME`* is the name of a job defined in the application and *`FILE_NAME`* is the name of the ZIP file (optional) into which data is to be exported.

Example

```
epmautomate exportData dailydataexport dailyData.zip
```

exportDataManagement

Exports Data Management records from an environment to a ZIP file.

This command exports a complete set of setup and staging table data including the ID columns to a ZIP file so that the data can be imported without losing referential integrity.

The exported file, for example, `dataFile.zip` is stored in the outbox. You can download the exported file using the downloadFile command, for example, `epmAutomate downloadFile outbox/dataFile.zip`. You can use this ZIP file to import the data using the importDataManagement command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate exportDataManagement FILE_NAME.zip` where *`FILE_NAME`* is the name of the ZIP file into which data is to be exported.

Example

```
epmautomate exportDataManagement dataFile.zip
```

exportDimension

Exports a dimension from an Oracle Enterprise Data Management Cloud application to a file in the staging area or, optionally, to a target environment defined in a connection.

Applies to

Oracle Enterprise Data Management Cloud

Required Roles

Service Administrator, User (with Data Manager permission)

Usage

`epmautomate exportDimension APPLICATION DIMENSION FILE_NAME [connection=NAME]`
where:

- `APPLICATION` is the name of an Oracle Enterprise Data Management Cloud application
- `DIMENSION` is the name of an application dimension
- `FILE_NAME` is the name of the file (CSV for export to a file or ZIP for export to Oracle Financials Cloud) for storing the exported data. If the connection parameter value is not set, this file is created in the staging area. You can download it to a local computer using the downloadFile command or copy it to another Oracle Enterprise Performance Management Cloud environment using the copyFileFromInstance command.
- `connection=NAME`, optionally, identifies a connection name (instance location) defined in Oracle Enterprise Data Management Cloud. If specified, the export file is uploaded to the target environment (inbox for EPM Cloud and default upload location for Oracle Financials Cloud).

> **Note:**
>
> The credentials specified in the connection definition must have the access rights to write to the target environment.

Examples

- Export to Oracle Enterprise Data Management Cloud staging area: `epmautomate exportDimension USOperations Entity EntityData.CSV`
- Export and upload to Oracle Financials Cloud: `epmautomate exportDimension USOperations Entity EntityData.zip Connection=ora_fusion_gl`
- Export and upload to target EPM Cloud inbox: `epmautomate exportDimension USOperations Entity EntityData.CSV Connection=EPM_cloud_pln`

exportDimensionMapping

Exports mapping rules of a specific Oracle Enterprise Data Management Cloud dimension for a location to create a mapping rule file and, optionally, uploads the exported file to the Data

Management inbox of another Oracle Enterprise Performance Management Cloud environment.

Applies to

Oracle Enterprise Data Management Cloud

Required Roles

Service Administrator, User (with Data Manager permission)

Usage

`epmautomate exportDimensionMapping` *`APPLICATION DIMENSION LOCATION FILE_NAME`* `[connection=`*`NAME`*`]` where:

- `APPLICATION` is the name of an Oracle Enterprise Data Management Cloud application
- `DIMENSION` is the name of an application dimension
- `LOCATION` is the specific location for which mapping rules should be exported.
- `FILE_NAME` is the name of the CSV file for storing the exported mappings. This file is created in the staging area if `connection` parameter is not set; you can download it to a local computer using the downloadFile command or use the copyFileFromInstance command to copy the file to another EPM Cloud environment.
- `connection=`*`NAME`*, optionally, identifies a connection name (instance location) defined in Oracle Enterprise Data Management Cloud. If specified, EPM Automate uploads the exported file to the default upload location of the target environment.

> **Note:**
>
> The credentials specified in the connection must have the access rights to write to the target environment.

Examples

- Export to the staging area: `epmautomate exportDimensionMapping USOperations Entity Loc1 Loc1Mappings.CSV`
- Export and upload to the inbox of the target EPM Cloud environment: `epmautomate exportDimensionMapping USOperations Entity Loc1 Loc1Mappings.CSV Connection=EPM_cloud_pln`

exportEJJournals

Exports Enterprise Journals that are ready to be posted from Financial Consolidation and Close to a ZIP file. This file can then be used to import journal data into an ERP system.

After exporting journals to an export file, this command updates the post status of each exported journal from `Ready To Post` to `Post In Progress`.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator

Usage

`epmautomate exportEJJournals FILE_NAME.zip [year=YEAR [period=PERIOD]]` where:

- *`FILE_NAME`* identifies a ZIP file into which journal export CSV files are to be archived. The command generates one CSV file (name format is `YEAR_PERIOD_JOURNALID_YYYYDDMMHHMMSS.csv`) for each journal and then zips them to create this ZIP file.
- *`YEAR`*, optionally, is the data collection year for which the journal data is to be exported. If not specified, data from all years is exported.
- *`PERIOD`*, optionally, is the data collection period from which journal data is to be exported. May be set only if a data collection year is specified. If a value is not specified, data from all periods is exported.

> **Note:**
>
> If *`YEAR`* and *`PERIOD`* are not specified, this command exports all journals that are in `Ready To Post` posting status across all years and periods.

Examples

- Export Journal data for all years and periods:
  `epmautomate exportEJJournals Journal_Export.zip`
- Export Journal data for a specific year:
  `epmautomate exportEJJournals Journal_Export.zip year=2020`
- Export Journal data for a specific year and period combination:
  `epmautomate exportEJJournals Journal_Export.zip year=2021 period=March`

exportEssbaseData

Exports data from an application cube (Oracle Essbase cube) to an archive. You can export just the level 0 data (ASO and BSO cubes) or all data in the cube (BSO cubes).

Use the exported archive to analyze Essbase data for patterns, for example to help improve performance.

Running a data export places the cube into read-only mode and prevents any write activity while the export operation is in progress.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportEssbaseData CUBE_NAME FILE_NAME [level=0|All]` where:

- *CUBE_NAME* identifies the cube from which data is to be exported.
- *FILE_NAME* is the name of a ZIP file that will contain the exported data. You can download this archive by running the downloadFile command.
- `level`, optionally, identifies the level of data to be exported. Default is `0`.
  - **ASO cubes:** Specify `0` to export level 0 data. You cannot use the `All` option.
  - **BSO cubes:** Specify `0` to export level 0 data or `All` to export all data.

Examples

- Export all data from a BSO cube:
  ```
  epmautomate exportEssbaseData Report1 Report1_all_data.zip level=All
  ```
- Export level 0 data from a cube:
  ```
  epmautomate exportEssbaseData Plan1 Plan1_lvl0_data.zip
  ```

exportJobConsole

Exports the job console records to a CSV file and creates an export ZIP file.

The first character in the output CSV file is the Byte Order Mark (BOM) character `\ufeff` followed by an encrypted application identifier enclosed in double quotes. CSV file header follows the application identifier.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportJobConsole FILE_NAME.zip [nDays=NUMBER_OF_DAYS] [jobtypes=JOB_TYPE] [jobStatusCodes=STATUS_CODE] [exportErrorDetails=true|false] [excludeApplicationId=true|false]`, where:

- *FILE_NAME* is the name of the ZIP file that will store the exported job console records. You use the downloadFile command to download this file from an environment.
- `nDays`, optionally, identifies the number of days for which job console records are to be exported. Possible values are: `all` (in all lower case)to export all available job console records, `1`, `2`, `7`, `30`, and `60`. Default is `7`.
- `jobTypes`, optionally, is a comma separated list of job codes for which console records should be exported. Default is `Rules`. Valid values are:
  - `all` (in all lower case)
  - `RULES`

  - RULESET
  - CLEAR_CELL_DETAILS
  - COPY_DATA
  - INVALID_INTERSECTION_RPT
  - COPY_VERSIONS
  - CONTENT_UPGRADE
  - PLAN_TYPE_MAP
  - IMPORT_DATA
  - EXPORT_DATA
  - EXPORT_METADATA
  - IMPORT_METADATA
  - CUBE_REFRESH
  - CLEAR_CUBE
  - ADMIN_MODE
  - COMPACT_CUBE
  - RESTRUCTURE_CUBE
  - MERGE_DATA_SLICES
  - OPTIMIZE_AGGREGATION
  - SECURITY_IMPORT
  - SECURITY_EXPORT
  - AUDIT_EXPORT
  - JOBCONSOLE_EXPORT
  - SORT_MEMBERS
  - SMART_PUSH
  - IMPORT_EXCHANGE_RATES
- `jobStatusCodes`, optionally, is a comma-separated list of job status codes for which records are to be exported. Default is `2` (Completed Successfully). Possible values are:
  - `all` (in all lower case) for all jobs in any status
  - `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
- *`exportErrorDetails`*, optionally, exports the details of jobs that failed or reported error to log files if this value is set to `true`. This error log file is included in the output ZIP file. Default is `false`. If this value is set to `true`, status details of jobs in the following status are exported.
  - Failed with errors
  - Completed with unknown status
  - Completed with threshold violation status
  - Completed with errors
  - Completed with warnings
- `excludeApplicationId`, optionally identifies whether the application identifier is to be written to the export file. Default is `false`.

> **Note:**
>
> Data from exported files that do not contain the application identifier cannot be imported into Oracle Enterprise Performance Management Cloud environments.

Examples

- Export all available job console records:
  ```
  epmautomate exportJobConsole jobs.zip nDays=all jobTypes=all
  jobStatusCodes=all
  ```
- Export all available Rules job console records:
  ```
  epmautomate exportJobConsole jobs.zip nDays=all jobStatusCodes=all
  ```
- Export all available Rules job console records without application identifier:
  ```
  epmautomate exportJobConsole jobs.zip nDays=all jobStatusCodes=all
  excludeApplicationId=true
  ```
- Export only the records for Rules job that finished successfully in the last 14 days:
  ```
  epmautomate exportJobConsole jobs.zip nDays=14
  ```
- Export console records and errors of import metadata and clear cube jobs that failed with errors or completed with errors run in the last seven days:
  ```
  epmautomate exportJobConsole jobs.zip jobtypes=IMPORT_METADATA,CLEAR_CUBE
  jobStatusCodes=3,8 exportErrorDetails=true
  ```

exportLibraryArtifact

Exports Narrative Reporting library artifacts. Optionally, for report artifacts only, this command can convert the export to an LCM file that you can import into Financial Consolidation and Close, Planning, Planning Modules, or Tax Reporting.

On completing the export, use the downloadFile command to download the export and error files to a local computer.

Applies to

Narrative Reporting

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer must be granted additional security through ACL

Usage

`epmautomate exportLibraryArtifact` *`ARTIFACT_PATH EXPORT_FILE`* `[exportFormat=Native|File|LCM] [applicationName=`*`APP_NAME`*`] [errorFile=`*`ERROR_FILE`*`.txt]` where:

- *`ARTIFACT_PATH`* is the location of the artifact in the Narrative Reporting library.
- *`EXPORT_FILE`* is a unique name for the file to which artifacts are to be exported.
- `exportFormat`, optionally, is one of the following:
  - `Native` exports artifacts as a zip file that can be used with other Narrative Reporting environments. This is the default value.
  - `File` exports files in the original binary format (PDF, DOCX, Zip, JPEG, and so on) in which they are available within Narrative Reporting. This parameter can be used to export binary files only; it should not be used with Reports artifacts.
  - `LCM` converts reports to the format used by Migration and exports them in a ZIP file that can be imported into Financial Consolidation and Close, Planning, Planning Modules, or Tax Reporting environments.
- `applicationName`, optionally, is the name of the target application into which you plan to import the reports. This value is required only if you are using `LCM` as the value of the `exportFormat` parameter.
- *`errorFile`*, optionally, is a unique name for the text file that will store export-related errors.

Examples

- Export a report in its native format so that it can be imported into another Narrative Reporting environment:
  ```
  epmautomate exportLibraryArtifact "Library/Samples/Sample Report 1"
  exp_SampleReport1.doc errorFile=export_errors.txt
  ```
- Export a spreadsheet in its original binary format:
  ```
  epmautomate exportLibraryArtifact "Library/Spreadsheets/Sheet1.xlsx"
  exp_Sheet1.xlsx exportFormat=File errorFile=export_errors.txt
  ```
- Export reports and format them for import into Financial Consolidation and Close, Planning, Planning Modules, or Tax Reporting:
  ```
  epmautomate exportLibraryArtifact "Library/Samples/Sample Report 1"
  exp_SampleReport1.zip exportFormat=LCM applicationName=Vision
  errorFile=report_exp_errors.txt
  ```

exportLibraryDocument

Exports any document available in the Reports library to a file.

You can download the exported file to a local computer using the downloadFile command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportLibraryDocument ARTIFACT_PATH [jobName=JOB_NAME] [exportFile=FILE_NAME] [exportFormat=file|zip] [errorFile=FILE_NAME.log] [overWrite=true|false]` where:

- *ARTIFACT_PATH* is the location of the document in the Reports library.
- *jobName*, optionally, is the name of the library artifact export job that is to be used to export the document. The default job name is `Copy Artifact From Library`.
- *EXPORT_FILE* is a unique name for the file to which document is to be exported. If you do not specify this value, the export file will be created using the name of the document in the library.
- `exportFormat`, optionally, is one of the following:
  - `File` exports the document in the original binary format (PDF, DOCX, Zip, JPEG, and so on) in which it is available within the library. This is the default value.

    > **Note:**
    >
    > In the 24.02 update, this option does not work.

  - `zip` exports a ZIP file containing the document in its original binary format. This is the only option that works in this update.
- *errorFile*, optionally, is a unique name for the file that will store export-related errors. No error file is created if you do not specify this value.
- `overwrite`, optionally, controls whether an identically named file currently in the default download location should be overwritten. The default is `false`, which means that the command will fail if a file with an identical name exists in the outbox.

Example

```
epmautomate exportLibraryDocument Library/folder1/WeeklySales.html jobName="Copy Weekly Sales" exportFile=WeeklySales.zip errorFile=WeeklySalesError.log overWrite=true exportFormat=zip
```

exportLogoImage

Exports the corporate logo used in an Account Reconciliation business process to a JPG file so that you can import it into another environment.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmautomate exportLogoImage IMAGE_NAME.jpg`, where *IMAGE_NAME* is the name for the logo image file.
You can download the exported logo file using the downloadFile command. Use the uploadFile command to upload it to the target environment, and then run the importLogoImage command.

Example

```
epmautomate exportLogoImage corpLogo.jpg
```

exportMapping

Exports mapping rules of a specific dimension or location to create a mapping rule file. You must specify the file name and a location within the inbox (for example, `inbox/exportedAccountMap.txt` or `inbox/france sales/exportedAccountMap.txt`) to export mappings.

Use the downloadFile command to download the exported mapping file to a local computer.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate exportMapping DIMENSION_NAME|ALL FILE_NAME LOCATION` where:

- *DIMENSION_NAME|ALL* is the source dimension from which mappings are to be exported. Specify the name of the dimension from which mappings are to be exported or `ALL` to export mappings from all dimensions of a location.
- *FILE_NAME* is a unique name for the mapping file and a location within the outbox.
- *LOCATION* is the Data Management location for which mapping rules should be exported.

Examples

- `epmautomate exportMapping Account inbox/exportedAccountMap.txt "France Sales"`
- `epmautomate exportMapping ALL "inbox/france sales/exportedAccountMap.txt" "France Sales"`

exportMetadata

Exports metadata into a file using the settings specified in a job of type `export metadata`. The file containing the exported data is stored in the default download location from where you can download it to a local computer.

Optionally, you can specify a file name for the exported data, which overrides the default file name (job name that is used to export metadata). Metadata is exported as a ZIP file only.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportMetadata` *`JOB_NAME [FILE_NAME]`* where *`JOB_NAME`* is the name of a job defined in the application and *`FILE_NAME`* is the name of the ZIP file into which metadata is to be exported.
Use the downloadFile command to download this file to a local server.

Example

```
epmautomate exportMetadata dailyAccountexport Accountexport.ZIP
```

exportOwnershipData

Exports ownership data from an entity to a comma-delimited CSV file.

Default ownership data that was populated by Financial Consolidation and Close is not included in the export file. Only the data that users entered to override the default settings is included in the export file.

Applies to

Financial Consolidation and Close and Tax Reporting.

Required Roles

Service Administrator, Power User, User

Usage

`epmautomate exportOwnershipData` *`Entity Scenario Year Period FILE_NAME`* where:

- `Entity` is the name of the entity from which data is to be exported.
- `Scenario` is the scenario from which data is to be exported.
- `Year` is the year from which data is to be exported.
- `Period` is the period of the year from which data is to be exported.
- `FILE_NAME` is the name of a CSV file to which the data is to be exported. Use the downloadFile command to download this file to a local server.

Example

```
epmautomate exportOwnershipData FCCS_TotalActual FY18 Dec exportfile.csv
```

exportQueryResults

Runs a query defined in an application and exports results into a text file.

The query result file is stored in profitoutbox; you can download it using the downloadFile command or by using Profitability and Cost Management File Explorer.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate exportQueryResults APPLICATION_NAME fileName=FILE_NAME [fileOutputOptions=ZIP_ONLY|ZIP_AND_TEXT|TEXT_ONLY] [queryName=QUERY_NAME] [exportOnlyLevel0Flg=true|false] [roundingPrecision=2] [dataFormat=NATIVE|COLUMNAR] [memberFilters=JSON_FILTER] [includeHeader=true|false] [delimiter="DELIMITER"] [keepDuplicateMemberFormat=true|false]` where:

- *APPLICATION_NAME* is the name of the Profitability and Cost Management application for which you want to run the query.
- `fileName` is the name of the file that will store the query results. This parameter value is required if `queryName` parameter value is not specified. It is optional if `queryName` parameter value is specified, in which case, the query name is used as the name of the query results file.
  The data format you specify determines the format of the output file. If you use `dataFormat=NATIVE` (default) the export process creates a text file. If you use `dataFormat=COLUMNAR`, the export process creates multiple sequentially numbered text files and compresses them into a Zip file.
- `fileOutputOptions`, optionally, identifies the output format of the query results file. Default is `ZIP_ONLY`, which creates `fileName`.ZIP or `queryName`.ZIP depending on whether a value for the `fileName` parameter is specified. Other options are `TEXT_ONLY` to create the output file as a text file and `ZIP_AND_TEXT` to generate both a text file and a zip file.
- `queryName` is an optional parameter that identifies a query that is defined in the application. Query names that contain the space character must be enclosed in double quotation marks.
  Do not specify a query name if you want to export all Oracle Essbase data belonging to the application.

  The following conditions may cause this command to create an empty data file:

  - A badly formed query that retrieves no data
  - A query that generates too much data. In this scenario, consider narrowing the scope of the query so that it retrieves less data or break the query into smaller queries
    See Managing Oracle Profitability and Cost Management Cloud Queries in *Administering Profitability and Cost Management*.
- `exportOnlyLevel0Flg`, optionally, specifies whether the query should retrieve only level0 data. Specify this parameter value in all lower case.
  This parameter is ignored if you are exporting all application data by omitting the query name.
- `roundingPrecision`, optionally, specifies the number of decimal places (rounding precision) to use when exporting query results. Applicable only when `queryName` is specified. Default is 2.
- `dataFormat`, optionally, identifies the output format. Valid values are:

  - `NATIVE`, which maintains the query result as Essbase native format data. This is the default value.
  - `COLUMNAR`, which converts Essbase native format data and orders it in columns for easy interpretation and import into other applications.
    This option exports all Essbase data and ignores the `queryName` parameter value. You can filter the data by setting the `memberFilters` parameter value.

> **Note:**
>
> The command considers the following optional parameters only if `dataFormat` is specified as `COLUMNAR`.

- `memberFilters`, optionally, accepts a JSON formatted string to filter by dimension and level0 members. Example, `"{\"Dim1\":[\"Mem1\"],\"Dim2\":[\"Mem21\",\"Mem22\"]}"`
- `includeHeader`, optionally, adds dimension names as column headers. Set this value to `false` to exclude column header. Default is `true`.
- `delimiter`, optionally, identifies the delimiter that is to be used to separate dimension members in the query result file. Delimiter must be enclosed in double quotation marks. Default is space (" ").
- `keepDuplicateMemberFormat`, optionally, specifies whether to print the member format in Essbase duplicate member format, for example, `[Account]@[Accoun1]`. Set this value to `false` to print only member name. Default is `true`.

Examples

- Export all application data:
```
epmautomate exportQueryResults BksML12 fileName="BksML12_MyQuery1.txt"
fileOutputOptions=TEXT_ONLY
```
- Export results of a specific query:
```
epmautomate exportQueryResults BksML12 queryName="My Product Query"
roundingPrecision=3
```
- Export Level0 data in NATIVE data format:
```
epmautomate exportQueryResults BksML30 fileName="BksML30_ExportLevel0-Data"
fileOutputOptions=ZIP_AND_TEXT exportOnlyLevel0Flg=true
```
- Export Level0 data in COLUMNAR data format with a single dimension and single member filter:
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Period\":[\"December\"]}"
includeHeader="true" delimiter="," roundingPrecision="3"
```
- Export Level0 data in COLUMNAR data format with a single dimension and multiple member filters:
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Period\":[\"November\",\"December\"]}"
includeHeader="true" delimiter="," roundingPrecision="3"
```
- Export Level0 data in COLUMNAR data format with a multiple dimensions and multiple member filters:
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Year\":[\"2016\"],\"Period\":
[\"November\",\"December\"]}" includeHeader="true" delimiter=","
roundingPrecision="3"
```

exportSnapshot

Repeats a previously performed export operation to create a snapshot of Migration content.

Using Migration, select and export desired artifacts to a snapshot; for example, `January16FullApp`. Use the snapshot name with this command to subsequently repeat the export operation, which will export only the artifacts that were selected during the original export operation. See Exporting Artifacts and Application in *Administering Migration for Oracle Enterprise Performance Management Cloud*.

- The following are not part of Planning, Planning Modules, and FreeForm application snapshots:
  - Audit data
  - Job Console data

  Use the cloneEnvironment command or the Clone Environment feature if you want to copy audit and Job Console data to the target environment.
- Snapshots do not contain the Data Management staging table data. To import this data, use the exportDataManagement and importDataManagement commands or the Data Management System Maintenances Scripts interface. You may use the cloneEnvironment command or the Clone Environment feature to create an identical copy of the environment, including the Data Management staging table data.

You can download the exported snapshot from the default location using the downloadFile command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate exportSnapshot` *`SNAPSHOT_NAME`* where *`SNAPSHOT_NAME`* is the name of an existing snapshot in Migration. This snapshot is replaced by the new snapshot.

Example

```
epmautomate exportSnapshot January16FullApp
```

exportTemplate

Exports an application as a template into a .ZIP file. The exported file is stored in profitoutbox.

You can download the exported file to a local computer using the downloadFile command.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate exportTemplate APPLICATION_NAME File_Name` where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that you want to export as template
- `File_Name` is the name for the template file

Example

```
epmautomate exportTemplate BksML12 template1
```

exportTaskManagerAccessControl

Exports the user details report for Task Manager, Supplemental Data, and Enterprise Journal user assignments in Financial Consolidation and Close and Tax Reporting. The 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, to an Excel or CSV file.

A sample Task Manager Access Control 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	Organizatic	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	user Ats	ats_user	Available		example1@example.com	User									User
5	user2 Ats	ats_user2	Available		example2@example.com	user									User
6	View User	viewUser	Available		view.user@exaple.com	Viewer									Viewer

Applies to

Financial Consolidation and Close and Tax Reporting

Required Roles

Service Administrator

Usage

`epmAutomate exportTaskManagerAccessControl REPORT_NAME` where *`REPORT_NAME`* is the name of the export file (including a valid (CSV or XLS) extension) that will contain the report.

This report can be generated in CSV or XSL format. You can download it using the downloadFile command.

Examples

- `epmAutomate exportTaskManagerAccessControl aclreport.csv`
- `epmAutomate exportTaskManagerAccessControl aclreport.xls`

exportValidIntersections

Exports valid intersection groups from the business process into a ZIP file, which you can download to a local computer using the downloadFile command. Valid intersections are cell interactions that are filtered based on rules, called valid intersection rules, that you define. These rules filter certain cell intersections to users when they enter data or select runtime prompts.

Applies To

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate exportValidIntersections FILE_NAME.zip [names=INTERSECTION_NAMES]` where:

- `FILE_NAME` is a name for the export ZIP file. All valid intersection identified in the command are exported to a Microsoft Excel file and then zipped to create this file.
- `names`, optionally, identifies a comma separated list of valid intersections that you want to export. If this parameter value is not specified, EPM Automate exports all valid intersections in the application.

Examples

- **Export specific valid intersections**
  ```
  epmautomate exportValidIntersections VI_export_File.zip
  names=VIAccountPeriod,VIEntityPeriod,VIProductPeriod
  ```
- **Export all valid intersections**
  ```
  epmautomate exportValidIntersections VI_export_File.zip
  ```

extractDimension

Extracts an Oracle Enterprise Data Management Cloud dimension to a file or to a global connection.

Applies to

Oracle Enterprise Data Management Cloud

Required Roles

Service Administrator, User (with Data Manager permission)

Usage

`epmautomate extractDimension APPLICATION DIMENSION EXTRACT_PROFILE FILE_NAME [connection=NAME]` where:

- `APPLICATION` is the name of an Oracle Enterprise Data Management Cloud application.

- `DIMENSION` is the name of the dimension to be extracted.
- `EXTRACT_PROFILE` is the name of an extract profile defined in the application. This profile is used to extract the dimension.
- `FILE_NAME` is the name of a file (CSV for export to a file or ZIP for export to Oracle Financials Cloud) for storing the extracted data. If the connection parameter value is not set, this file is created in the staging area. You can download it to a local computer using the downloadFile command or copy it to another Oracle Enterprise Data Management Cloud environment using the copyFileFromInstance command.
- `connection=`*`NAME`* optionally, identifies a global connection name (instance location) defined in Oracle Enterprise Data Management Cloud as the location of the file. If specified, the extract file is uploaded to the target environment (inbox for Oracle Enterprise Performance Management Cloud and the specified document account for Oracle ERP).

> **Note:**
>
> The credentials specified in the global connection must have access rights to write to the target environment.

Examples

- Extract to Oracle Enterprise Data Management Cloud staging area: `epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.CSV`
- Extract and upload to Oracle ERP: `epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.zip Connection=ora_fusion_gl`
- Extract and upload to target EPM Cloud inbox: `epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.CSV Connection=EPM_cloud_pln`

extractPackage

Runs an extract package consisting of multiple extracts for an application using a single operation. The results of each extract in the package are added to one ZIP file, which can be written to the staging area or to a global connection.

Applies to

Oracle Enterprise Data Management Cloud

Required Roles

Service Administrator, User (with Data Manager permission)

Usage

`epmautomate extractPackage` *`APPLICATION_NAME EXTRACT_PACKAGE_NAME FILE_NAME`*`.zip [connection=`*`CONNECTION_NAME`*`]`, where:

- *`APPLICATION_NAME`* is the name of an Oracle Enterprise Data Management Cloud application.
- *`EXTRACT_PACKAGE_NAME`* is the name of the package defined in the Oracle Enterprise Data Management Cloud application.

- *FILE_NAME* is the name for the extract ZIP file that will contain the results of the extraction. This file is created in the staging area or written to a global connection if a connection name is specified.
  A package may contain one or more extract profiles, each with a distinct name. The extract package ZIP file will contain the output files of the extracts in the package. The number of files in the ZIP file is determined by the settings in the package.
- *CONNECTION_NAME*, optionally, is the global connection name (instance location) defined in Oracle Enterprise Data Management Cloud. It identifies the location to which the ZIP file containing the results of the extraction is to be uploaded.
  If a connection name is specified, the ZIP file containing extracted data is uploaded to the target environment (inbox for an Oracle Enterprise Performance Management Cloud environment or the specified document account for Oracle ERP). If a value is not specified, the export file is created in the default outbox (staging area) of the Oracle Enterprise Data Management Cloud environment.

Examples

- Extract COARedesignMaps package to a file in the Oracle Enterprise Data Management Cloud staging area:

```
epmautomate extractPackage FinancialsCloud COARedesignMaps COARedesign.zip
```

- Extract COARedesignMaps package to a file and upload it to a Oracle Fusion Cloud EPM inbox using a connection.

```
epmautomate extractPackage CorporateAccount COARedesignMaps
COARedesign.zip Connection=EPM_cloud_pln
```

feedback

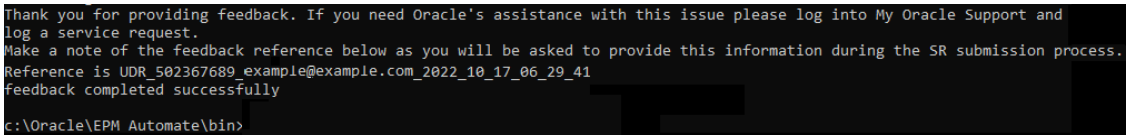
Sends feedback to Oracle and to the Service Administrators of the environment and automatically uploads all the EPM Automate log files created in the last 24 hours from the current directory.

You may, optionally, upload additional files (for example Fiddler trace files) that you may want Oracle Support to use to diagnose why the current issue occurs.

This command, which mimics the Provide Feedback feature of the service, is especially useful for providing feedback to Oracle in cases where the user interface is unresponsive or you encounter an issue while running EPM Automate.

For information on the Provide Feedback feature, see Helping Oracle Collect Diagnostic Information Using the Provide Feedback Utility in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

This command returns a message, similar to the following, informing that feedback does not create a service request. If you need help from Oracle to resolve an issue, you must file a service request. The command displays a UDR reference number, which you should include in the service request that you file.



Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate feedback "Comment" [Screenshot="FILE_PATH"] [File="FILE_PATH"]` where:

- `Comment` is text that describes the issue about which this feedback is being submitted. Comments must be enclosed in quotation marks.
- `Screenshot`, optionally, identifies the name of a graphic file that illustrates the issue for which this feedback is being submitted. You can submit multiple screenshots by repeating this parameter and value as needed.
- `File`, optionally, identifies the name of a file that you want Oracle support to use to resolve the current issue. Use this parameter to submit Fiddler traces or other files to Oracle. You can submit multiple files by repeating this parameter and value as needed.

Examples

- Windows: `epmautomate Feedback "runplantypemap CampaignToReporting ClearData=True did not clear data from aggregate storage" Screenshot=C:/feedback/issue.jpg File=exampleScript.ps1 file=trace.har`
- Linux: `epmautomate Feedback "runplantypemap CampaignToReporting ClearData=True did not clear data from aggregate storage" Screenshot=/scratch/screens/issue.jpg File=/home/feedback/trace.har`

getApplicationAdminMode

Checks whether the application is in administration mode with access limited only to Service Administrators.

This command, which returns `true` if the application is in administration mode and `false` otherwise, is useful for checking the status of the application before running automation scripts. For example, the refreshCube command requires the application to be in administration mode. You can use this command in the automation script as follows to check if the application is in administration mode.

```
adminMode = 'epmautomate.sh getApplicationAdminMode'
        if ["$adminMode" == "true"]
              epmautomate.sh refreshCube
```

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Account Reconciliation, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

```
epmautomate getApplicationAdminMode
```

Example

```
epmautomate getApplicationAdminMode
```

getDailyMaintenanceStartTime

Displays, in the console, the Coordinated Universal Time (UTC) or, optionally, the time zone, at which the daily maintenance of the environment is scheduled to start.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate getDailyMaintenanceStartTime [timezone=true|false]` where `timezone=true`, optionally, identifies whether to display the daily maintenance start time in the time zone specified while setting it, for example, `America/Los_Angeles`. Default is `false`.

Examples

- Display the maintenance time in the time zone specified while setting it:
  `epmautomate getDailyMaintenanceStartTime timezone=true`
- Display the maintenance time in UTC:
  `epmautomate getDailyMaintenanceStartTime`

getEssbaseQryGovExecTime

Displays the current maximum amount of time, in seconds, that an Oracle Essbase query can use to retrieve and deliver information before the query is terminated.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Profitability and Cost Management, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

```
epmautomate getEssbaseQryGovExecTime
```

A sample command output:

```
c:\Oracle\EPM Automate\bin>epmautomate getEssbaseQryGovExecTime
300

c:\Oracle\EPM Automate\bin>epmautomate setEssbaseQryGovExecTime 600
setEssbaseQryGovExecTime completed successfully


c:\Oracle\EPM Automate\bin>epmautomate getEssbaseQryGovExecTime
600
```

Example

```
epmautomate getEssbaseQryGovExecTime
```

getIdleSessionTimeout

Displays the session timeout (in minutes) of the Oracle Enterprise Performance Management Cloud environment. After a session is idle for this duration, users are redirected to the Login page.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

```
epmautomate getIdleSessionTimeout
```

A sample command output:

```
c:\Oracle\EPM Automate\bin>epmautomate getIdleSessionTimeout
75
```

getIPAllowlist

For OCI (Gen 2) environments, displays the IP addresses and Classless Inter-Domain Routings (CIDRs) included in the current allowlist.

This command is useful in checking whether a specific IP address or CIDR is currently permitted to access an OCI (Gen 2) environment.

> **Note:**
>
> This command cannot be used to list IP addresses and CIDRs in Classic environments. For Classic environments, use the Service Details screen of My Services (Classic) to work with the allowlist or denylist rules.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

```
epmAutomate getIPAllowlist
```

> **Note:**
>
> To write all existing IP addresses and CIDRs to a file, redirect the output to a text file, which you can edit (remove a few or all IP addresses and CIDRs), upload to the environment, and then use the setIPAllowlist command to remove the entries in the file from the allowlist. Command execution example:
>
> ```
> epmAutomate getIPAllowlist > myRemoveList.txt
> epmAutomate uploadFile myRemoveList.txt
> epmAutomate setIPAllowlist remove myRemoveList.txt
> ```

Example

Display the IP addresses and CIDRs included in the current allowlist:

```
epmAutomate getIPAllowlist
```

getRestrictedDataAccess

Displays whether the environment is configured to prevent Service Administrators from consenting to submit an application snapshot to Oracle while using the Provide Feedback utility.

This command reports `true` if the environment is configured to prevent Service Administrators from consenting to submit the application snapshot and `false` otherwise

Applies To

Account Reconciliation, Oracle Enterprise Data Management Cloud, Enterprise Profitability and Cost Management, Financial Consolidation and Close, FreeForm, Planning, Planning

Modules, Profitability and Cost Management, Narrative Reporting, Sales Planning, Strategic Workforce Planning, and Tax Reporting.

Required Roles

Service Administrator

Usage

To change the current data access restriction status, use setRestrictedDataAccess.

Example

```
epmautomate getRestrictedDataAccess
```

getSubstVar

Retrieves the values of substitution variables and displays them on screen.

The display format is *CUBE_NAME*.*SUBSTVAR*=value, for example, Plan2.CurYear=2016. Application level substitution variable values are displayed in *ALL*.*SUBSTVAR*=value format, for example, ALL.CurYear=2016

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User (with Rule Launch access)

Usage

epmautomate getSubstVar *CUBE_NAME*|ALL [*name=VARIABLE_NAME*] where:

- *CUBE_NAME* is the cube (for example, Plan1, Plan2) from which you want to retrieve the substitution variable. Use ALL to retrieve substitution variables at the application level.
- *name=VARIABLE_NAME* optionally identifies the substitution variable for which you want to retrieve value. If you do not specify a variable name, the command retrieves the value of all substitution variables.

Examples

- Get the value of all substitution variables at the application and cube level: epmautomate getSubstVar ALL
- Get the value of one specific substitution variable at the application level: epmautomate getSubstVar ALL name=CurYear
- Get the value of all substitution variables at the cube level: epmautomate getSubstVar Plan2
- Get the value of a specific substitution variable at the cube level: epmautomate getSubstVar Plan2 name=CurYear

getVirusScanOnFileUploads

Checks whether your OCI (Gen 2) environment is enabled to scan all files being uploaded to ensure that they are virus free.

This command checks whether virus scanning is enforced before files are uploaded to the environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

```
epmautomate getVirusScanOnFileUploads
```

This command prints `true` if the environment is enabled to scan uploaded files for virus; else it prints `false`.

groupAssignmentAuditReport

Creates a report listing users and groups that were added to or removed from Access Control groups for a specified date range.

This report, which is generated as a CSV file, can be used to support security audit operations. Each row of the generated CSV file 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. This report does not contain audit information on when groups were added to or deleted from Access Control.

	A	B	C	D	E
1	**User/Group Name**	**Group**	**Action**	**Administrator**	**Date and Time**
2	testGroup1	exampleGroup1	Added	john.smith@example.com	May 10, 2022 23:13:20 UTC
3	johndoe@example.com	exampleGroup1	Added	joe.doe@example.com	May 11, 2022 23:14:20 UTC
4	JaneDoe@example.com	testGroup1	Added	joe.doe@example.com	May 11, 2022 23:14:50 UTC
5	jaDoe@example.com	testGroup1	Added	john.smith@example.com	May 12, 2022 23:25:20 UTC
6	JaneDoe@example.com	testGroup1	Removed	john.smith@example.com	May 13, 2022 18:13:20 UTC
7	jaDoe@example.com	testGroup1	Removed	john.smith@example.com	May 13, 2022 18:22:20 UTC
8	testGroup1	exampleGroup1	Removed	joe.doe@example.com	May 13, 2022 23:13:20 UTC

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmAutomate groupAssignmentAuditReport` *`FROM_DATE TO_DATE REPORT_NAME`* where

- *`FROM_DATE`* is the start date of the period, in `YYYY-MM-DD` format, for which the report is to be generated.
- *`TO_DATE`* is the end date of the period , in `YYYY-MM-DD` format, for which the report is to be generated.
- *`REPORT_NAME`* is the name of a CSV file for the report. You can download the generated report using the downloadFile command.

Example

```
epmAutomate groupAssignmentAuditReport 2022-03-01 2022-05-01
GroupAssignmentReport.CSV
```

help

Displays help for all EPM Automate commands.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

```
epmautomate help
```

Example

```
epmautomate help
```

importAppAudit

Imports data audit records from a ZIP file that you created by exporting audit data from an environment.

You create the import file using the exportAppAudit command (`epmautomate exportAppAudit auditData ndays=All`). Use this command to clone audit records from one environment to another during migration or cloning for disaster recovery.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate importAppAudit FILE_NAME [logFilename=LOG_FILE_NAME]` where:

- `FILE_NAME` is the name of a ZIP file containing data audit records that you want to import into the application. Before running this command, use the uploadFile command to upload this file to the environment.
- `logFileName`, optionally, identifies the error log file in which errors encountered during the import will be recorded. If this value is not specified, the command generates an error file which is named using this convention: `username_date_timestamp`. You can download this file using the downloadFile command.

Example

```
epmautomate importAppaudit Audit_data.zip logFileName=auditImportLog
```

importAppSecurity

Loads access permissions for users or groups of an application from a CSV file available in the inbox.

Importing access permissions overwrites existing assignments only for imported members, data forms, data form folders, task lists, Calculation Manager business rules, and business rule folders. All other existing access permissions remain intact.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate importAppSecurity ACL_FILE_NAME ERROR_FILE [clearall=true|false]` where:

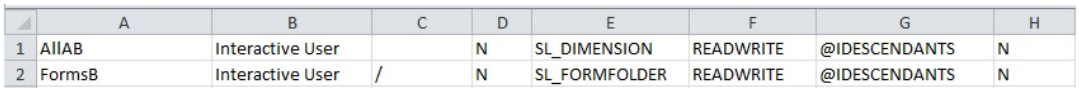
- `ACL_FILE_NAME` is the name of a CSV file containing access permissions that you want to import into the application. Before running this command, use the uploadFile command to upload this file to the inbox. Contents of a sample input file may be as shown in the following image:

	A	B	C	D	E	F	G	H
1	Object Name	Name	Parent	Is User	Object Type	Access Type	Access Mode	Remove
2	AllAB	Interactive User		N	SL_DIMENSION	READWRITE	@IDESCENDANTS	N
3	FormsB	Interactive User	/	N	SL_FORMFOLDER	READWRITE	@IDESCENDANTS	N
4	Statistics	Interactive User		N	SL_DIMENSION	READWRITE	@IDESCENDANTS	N
5	TD	Interactive User		N	SL_DIMENSION	READWRITE	@IDESCENDANTS	N
6	No Entity	Interactive User		N	SL_DIMENSION	READWRITE	MEMBER	N

  For a description of the column headers and possible values, see Import Security in *REST API for Oracle Enterprise Performance Management Cloud*.

- `ERROR_FILE` is the name of a CSV file, which EPM Automate will create in the outbox, to record the errors that are detected during this operation. You can download this file to a

local computer to analyze and correct the reported errors. Contents of a sample error file may be as shown in the following image. The columns of this file corresponds to the header columns of the input file:



	A	B	C	D	E	F	G	H
1	AllAB	Interactive User		N	SL_DIMENSION	READWRITE	@IDESCENDANTS	N
2	FormsB	Interactive User	/	N	SL_FORMFOLDER	READWRITE	@IDESCENDANTS	N

- `clearall`, optionally, specifies whether to delete the existing access permissions before loading new permissions from the file. Default is `false`.

Example

```
epmautomate importAppSecurity Acl_file.CSV Acl_import_error.CSV clearall=true
```

importARApplicationProperties

Imports application settings (Redwood Experience, theme, email notification, and business process name), logo, and background images available in an export JSON file into an Account Reconciliation environment.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmautomate importARApplicationProperties` *`FILE_NAME`* where *`FILE_NAME`* is the name of the JSON file exported from an environment.

This file, exported from another environment using the exportARApplicationProperties command, must be available in the environment where you are restoring application settings.

Example

```
epmautomate importARApplicationProperties myProp.JSON
```

importBackgroundImage

Imports the background image from an export file into an Account Reconciliation environment.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmautomate importBackgroundImage` *`FILE_NAME`*`.jpg`, where *`FILE_NAME`* is the name of the background image file exported from another environment.

Example

```
epmautomate importBackgroundImage image_file.jpg
```

importBalances

Uses Data Management to import balances data from a data load definition.

Applies to

Account Reconciliation.

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer must be granted additional security through ACL

Usage

`epmautomate importBalances` *`DL_DEFINITION PERIOD`* where:

- *`DL_DEFINITION`* is an existing data load definition in Account Reconciliation.
- *`PERIOD`* is the name of a period.

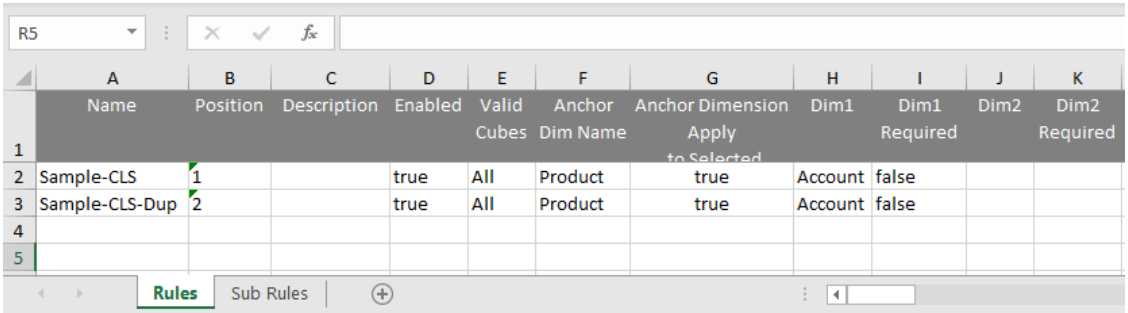
Example

```
epmautomate importBalances DailyLoad "January 2020"
```

importCellLevelSecurity

Imports cell-level security settings from a ZIP file that contains one Excel file with cell-level security records into the business process. Before running this command, use the uploadFile command to upload the import file to the environment.

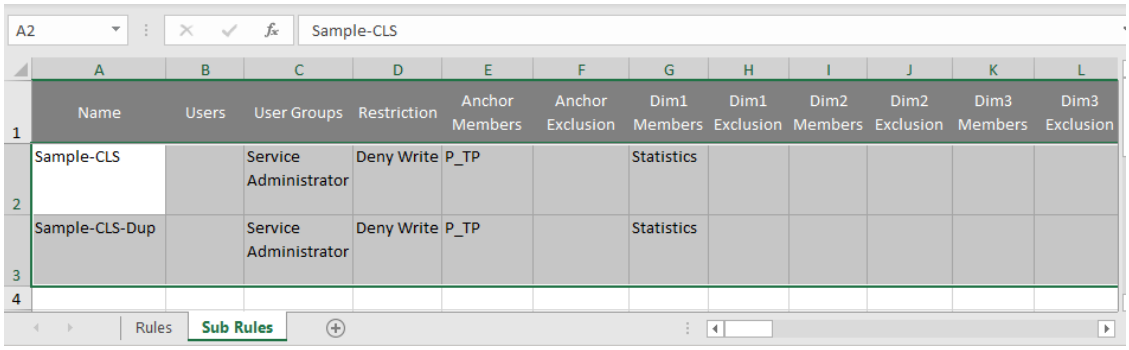
Your import ZIP file should contain one Excel file with two worksheets (Rules and Sub Rules) for successfully importing cell-level security. The Rules sheet should contain cell-level security definitions, dimensions included, properties such as Unspecified Valid and Additional Dims Required. The Sub Rules sheet should contain member selections and exclusions. The best method to get the import file format template is to export cell-level security from the application. A sample format is presented in the following illustrations.



R5

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Position	Description	Enabled	Valid Cubes	Anchor Dim Name	Anchor Dimension Apply to Selected	Dim1	Dim1 Required	Dim2	Dim2 Required
2	Sample-CLS	1		true	All	Product	true	Account	false		
3	Sample-CLS-Dup	2		true	All	Product	true	Account	false		
4											
5											

Rules | Sub Rules



Applies to

Planning, Planning Modules, FreeForm, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate importCellLevelSecurity` *`FILE_NAME`*`.ZIP [ErrorFile=`*`FILE_NAME`*`.txt]` where:

- `FILE_NAME` is the name of the ZIP file that contains Excel file with cell-level security information.
- `ErrorFile`, optionally, identifies the name of the text file to which error records will be written. If this parameter value is not specified, EPM Automate automatically generates an error file; you can view its name in the Job Console.
  Use the downloadFile command to download the error file to a local computer.

Example

```
epmautomate importCellLevelSecurity ImportCLSDRecordsFile.zip
ErrorFile=ImportCLSDRecords_errors.txt
```

importConsolidationJournals

Imports consolidation journals from a .JLF file into Financial Consolidation and Close.

- Use the exportConsolidationJournals command to create the .JLF file that is used as an input for this command.
- Before running this command, use the uploadFile command to load the input file into the environment.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator

Usage

`epmautomate importConsolidationJournals jobName [fileName=FILE_NAME] [errorFileName=ERROR_FILE_NAME]` where

- `jobName` is the name of an Import Journal job created in Financial Consolidation and Close.
- `fileName`, optionally, is the name of a .JLF file from which the journal entries are to be imported.
- `errorFileName`, optionally, is the name of the log file in which messages generated during the import process are to be recorded.

Example

```
epmautomate importConsolidationJournals "JIMPORT1" fileName="TestImport1.jlf"
errorFileName="TestImport1_error.log"
```

importData

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

Use the uploadFile command to upload the file containing application data to the default upload location.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate importData JOB_NAME [FILE_NAME] errorFile=ERROR_FILE.zip` where:

- *JOB_NAME* is the name of a job defined in the application.
- *FILE_NAME*, optionally identifies 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 identifies the import sequence in the file names; for example, `data1-3.csv`, `data2-3.csv`, and `data3-3.csv`.
- `errorFile`, optionally, identifies the name of a ZIP file in which rejected records, if any, during the import operations will be recorded. Identically named ZIP file in the outbox, if any, will be overwritten. You can download this file using the downloadFile command.

Example

```
epmautomate importData dailydataload dailydata.zip errorFile=dataImport_error.zip
```

importDataManagement

Imports Data Management records from a ZIP file into an environment.

This command imports data into setup and staging tables from a ZIP file created using the exportDataManagement command. Use the uploadFile command, for example, `epmAutomate uploadFile "C:/datafile/datafile.zip" inbox` to upload the import ZIP file to the Data Management inbox or to a folder within it.

> **Note:**
>
> This command can only import the Data Management records exported from another environment running on the same monthly update. For example, records exported from a 21.11 Oracle Enterprise Performance Management Cloud environment can only be imported into another 21.11 environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Account Reconciliation, Tax Reporting, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate importDataManagement` *`FILE_NAME`*`.zip` where *`FILE_NAME`* is the name of the ZIP file that contains the Data Management data to be imported.

Examples

- Import from the Data Management inbox:
  `epmautomate importDataManagement inbox/dataFile.zip`
- Import from a folder within inbox:
  `epmautomate importDataManagement inbox/dm_data/dataFile.zip`

importDimension

Imports a dimension from a file into an Oracle Enterprise Data Management Cloud application.

This command can import an input file from a connection defined in Oracle Enterprise Data Management Cloud or the staging area.

If the file is to be imported from the Oracle Enterprise Data Management Cloud staging area, you must use the uploadFile command to upload it into the target Oracle Enterprise Data Management Cloud environment. You may also use the copyFileFromInstance command to copy the file from another Oracle Enterprise Performance Management Cloud environment.

Applies to

Oracle Enterprise Data Management Cloud.

Required Roles

Service Administrator, User (with Data Manager permission)

Usage

`epmautomate importDimension APPLICATION DIMENSION IMPORT_TYPE FILE_NAME [connection=NAME]` where:

- `APPLICATION` is the name of an Oracle Enterprise Data Management Cloud application
- `DIMENSION` is the name of the application dimension being imported
- `IMPORT_TYPE` indicates how to perform the import. Valid import types are:
  - `ResetDimension` to delete all existing dimension data and import the new data
  - `ReplaceNodes` to add or update nodes and replace existing hierarchies during import
  - `Merge` to process incremental changes to the nodes and hierarchies using an import request
- `FILE_NAME` is the name of the file (CSV or ZIP) containing the dimension data to be imported. The file name must end with the dimension name prefixed with _ (underscore character); for example, `import_Entity.csv`. If you are importing from a ZIP file containing multiple import files, this command depends on the file name within the ZIP file to identify the right import file.
  If a value for `connection` is specified, you must import dimension from a a ZIP file; for example, `importdata_Entity.zip`.
- `connection=NAME`, optionally, identifies a connection name (instance location) defined in Oracle Enterprise Data Management Cloud as the location of the import file. If not specified, the import process will look in the local staging area for the import file.

Examples

- Import from a file uploaded to the staging area: `epmautomate importDimension USOperations Entity ReplaceNodes data_Entity.CSV`
- Import from the outbox of another EPM Cloud environment: `epmautomate importDimension USOperations Entity ReplaceNodes data_Entity.ZIP Connection=EPM_Cloud_pln`

importJobConsole

Clones job console records using a ZIP file containing job console records exported from an environment.

Importing job console records using this command is a one-time task that should be performed after running the recreate command. If you already used this command to import job console records, subsequent invocations of the command will fail until you recreate the environment.

Use the exportJobConsole command (`epmAutomate exportJobConsole FILE_NAME.zip nDays=All jobTypes=All jobStatusCode=All`) to create the ZIP file that is used as the input for this command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate importJobConsole FILE_NAME.zip [logFileName=jobConsoleLog]` where:

- *`FILE_NAME`* is the name of the ZIP file that contains the job console records that you want to import. You use the uploadFile command to upload this file to the environment.
- `logFileName`, optionally, identifies `jobConsoleLog` as the log file in which errors encountered during the import will be recorded. If this value is not specified, the command generates an error file which is named using this convention: *`username`*`importLog_date_timestamp.zip`. You can download this file using the downloadFile command.

Example

```
epmautomate importJobConsole jobConsole.zip jobConsoleLog
```

importLibraryArtifact

Imports library artifacts from an archive or file into Narrative Reporting library.

Before running this command, upload the source archive or file to the environment using the uploadFile command.

Applies to

Narrative Reporting

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer must be granted additional security through ACL

Usage

`epmautomate importLibraryArtifact SOURCE_FILE [errorFile=ERROR_FILE.txt] [importFormat=Native|File] [importFolder=FOLDER_PATH] [ importPermission=true|false] [overwrite=true|false]` where:

- *`SOURCE_FILE`* is the name of the archive that contains the artifacts that are to be imported into the library. This file must be available in the inbox.
- *`errorFile`*, optionally, is a unique name for the text file that will store import-related errors.
- `importFormat`, optionally, is one of the following:
  - `Native` imports artifacts from a zip file created using the exportLibraryArtifactcommand with the `exportFormat=Native` option. This is the default value.
  - `File` imports a binary file.

> **Note:**
>
> You use the importSnapshot command to import library artifacts zip files (created using the exportLibraryArtifactcommand with the `exportFormat=LCM` option) into Financial Consolidation and Close, Planning, Planning Modules, or Tax Reporting environments.

- `importFolder`, optionally, is the library location that will store the imported artifacts. Specify this path if this location is different than the `Library` (the default import location).
- *`importPermission`* indicates whether to import access permissions set for the artifacts. Default is `False`.
- *`overwrite`* identifies whether to overwrite identically named artifacts, if any, in the specified library location. Default is `False`, which means that the process will not import an artifact if an identically named artifact exists in the import location.

On completing the import, use the downloadFile command to download the error files to a local computer.

Examples

- Import a file in its binary format:
```
epmautomate importLibraryArtifact newReports.doc
errorFile=report_imp_errors.txt importFormat=File importFolder="Library/My
Reports" importPermission=true overwrite=true
```
- Import artifacts from an exported zip file:
```
epmautomate importLibraryArtifact newReports.zip
errorFile=report_imp_errors.txt importFormat=Native importFolder="Library/My
Reports" importPermission=true overwrite=true
```
- Import reports into a Financial Consolidation and Close, Planning, Planning Modules, or Tax Reporting environment from an exported zip file:
```
epmautomate importSnapshot newReports.zip
```

importLogoImage

Imports the corporate logo used in an Account Reconciliation environment from an export file into another environment.

Applies to

Account Reconciliation

Required Roles

Service Administrator

Usage

`epmautomate importLogoImage` *`IMAGE_NAME`*`.jpg`, where *`IMAGE_NAME`* is the name for the logo image file.
You can download the exported imageusing the downloadFile command. Upload it to the target environment using the uploadFilecommand, and then run the importLogoImage command to import it.

Example

```
epmautomate importLogoImage corpLogo.jpg
```

importMapping

Imports mappings from a mapping import file, which was previously uploaded to the environment.

Use the uploadFile command to upload files into Data Management inbox or a folder within it.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate importMapping` *`DIMENSION_NAME|ALL FILE_NAME IMPORT_MODE VALIDATION_MODE LOCATION`* where:

- *`DIMENSION_NAME|ALL`* indicates the recipient of the mapping. Specify the name of the dimension into which mappings are to be imported or `ALL` to import all mappings included in the file to appropriate dimensions.
- *`FILE_NAME`* is the name and location of the mapping import file available in Data Management inbox or a directory within it. Specify the file name (TXT files in standard Data Management format) and its path (for example, `inbox/AccountMap.txt` or `inbox/pbcs_maps/AccountMap.txt`).
- *`IMPORT_MODE`* is either `REPLACE` to clear existing mapping rules before importing mappings or `MERGE` to add new mapping rules to exiting rules.
- *`VALIDATION_MODE`* is `TRUE` to validate target members against the application or `FALSE` to load the mapping file without running validations.
- *`LOCATION`* is the Data Management location for which mapping rules should be loaded.

Examples

- `epmautomate importMapping Account inbox/AccountMap.txt MERGE FALSE "France Sales"`
- `epmautomate importMapping ALL "inbox/France Sales/AllMaps.txt" MERGE FALSE "France Sales"` (loads mappings from the mapping import file into all mapped dimensions in France Sales location)

importMetadata

Imports metadata into the application using the import settings specified in a job of type `import metadata`. Optionally, you can specify the name of the ZIP file from which metadata is to be imported.

Use the uploadFile command to upload the file containing the metadata to the default upload location.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate importMetadata` *`JOB_NAME [FILE_NAME]`* `errorFile=`*`ERROR_FILE`*`.zip` where:

- *`JOB_NAME`* is the name of a job defined in the application.
- *`FILE_NAME`*, optionally, identifies the name of the ZIP file from which metadata is to be imported. If specified, the contents of this ZIP file 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 *`_DIMENSIONNAME`*`.csv`; for example, `metadata_Entity.csv`, `metadata_HSP_Smart Lists.csv`, and `metadata_Exchange Rates.csv`.
- `errorFile`, optionally, identifies the name of a ZIP file in which rejected records, if any, during the import operations will be recorded. Identically named ZIP file in the outbox, if any, will be overwritten. You can download this file using the downloadFile command.

Note:

- You cannot rename members by running an import metadata job with a load file in which `old_name` or `unique_name` properties are modified. Renaming of members will be ignored.
- You cannot delete attribute dimensions while importing metadata using this command.
- Only the 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, are ignored.

  An ambiguous import situation is created if both of the following conditions are true for the ZIP file:

  – Zip contains a metadata file with a name that matches the file name defined in the job

  – Zip contains a metadata file or files with names that end in `_DIMENSIONNAME.CSV` or `_DIMENSIONNAME.TXT`, where *DIMENSIONNAME* is the name of the dimension into which metadata is being imported.

Oracle recommends that the ZIP file contains a metadata file with a name identical to that referenced in the job or a file with a name that ends in `_DIMENSIONNAME.CSV` (or `_DIMENSIONNAME.TXT`), but not both. For example, if you are loading a job that references the metadata file `Employees_A-Z.CSV` into the `Employees` dimension, your ZIP file may include `Employees_A-Z.CSV` or `New_Employees.CSV`, but not both. If your ZIP contains `Employees_A-Z.CSV` and `New_Employees.CSV`, EPM Automate may select either file for import depending on the order of the files in ZIP. `Employees_A-Z.CSV` file is a possible match for import because its name matches the file name referenced in the job; `New_Employees.CSV` is also a possible match because its name matches the `_DIMENSIONNAME.CSV` pattern.

Example

```
epmautomate importMetadata importAccount importAccount.zip
errorFile=metadataImport_error.zip
```

importOwnershipData

Imports ownership data from a CSV file available in the environment into a period.

Before running this command, use the uploadFile command to load the import source CSV file into the environment.

Header of this CSV file is as follows:

```
Scenario, Year, Period, Entity, Parent, POwn, Control, Method
```

`POwn`, `Control`, and `Method` values are optional.

The imported ownership data is merged with any existing data, which may create invalid ownership entries. If an entity is present in more than one branch of a hierarchy, the imported ownership data may cause the combined ownership % of the entity to exceed 100%. You must manually correct the ownership % to ensure that it does not exceed 100%.

Applies to

Financial Consolidation and Close and Tax Reporting.

Required Roles

Service Administrator, Power User, User
Users must have write access to the entity.

Usage

`epmautomate importOwnershipData Scenario Year Period FILE_NAME` where:

- `Scenario` is the scenario into which ownership data is to be imported.
- `Year` is the year into which data is to be imported.
- `Period` is the period of the year into which the ownership data is to be imported.
- `FILE_NAME` is the name of the CSV file from which data is to be imported.

Example

```
epmautomate importOwnershipData FCCS_TotalActual FY19 Jan importfile.csv
```

importPreMappedBalances

Imports pre-mapped balance data from a file in the Account Reconciliation repository.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importPreMappedBalances PERIOD FILE_NAME BALANCE_TYPE CURRENCY_BUCKET` where:

- *PERIOD* is the name of a period
- *FILE_NAME* is the name of the CSV file containing the data to be imported
- *BALANCE_TYPE* is `SRC` or `SUB`
- *CURRENCY_BUCKET* is `Entered`, `Functional` or `Reporting`

Example

```
epmautomate importPreMappedBalances "January 2015" dailydata.csv SRC Reporting
```

importPreMappedTransactions

Imports pre-mapped transactions from a CSV file in the Account Reconciliation repository.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importPreMappedTransactions` *`PERIOD TRANSACTION_TYPE FILE_NAME DATE_FORMAT`* where:

- *`PERIOD`* is the name of a period
- *`TRANSACTION_TYPE`* is one of the following:
  - `BEX` for loading Balance Explanations
  - `SRC` for loading Source System Adjustments
  - `SUB` for loading Subsystem Adjustments
  - `VEX` for loading Variance Analysis Explanations
- *`FILE_NAME`* is the name of the CSV file from which data is to be imported
- *`DATE_FORMAT`* is date format text string; for example, MMM d, yyyy.

Example

```
epmautomate importPreMappedTransactions "January 2015" "BEX" transactions.csv
"MMM d, yyyy"
```

importProfiles

Imports new profile definitions from a CSV file in the Account Reconciliation repository.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importProfiles` *`FILE_NAME PROFILE_TYPE METHOD DATE_FORMAT`* where:

- *`FILE_NAME`* is the name of the CSV file from which data is to be imported
- *`PROFILE_TYPE`* is either `profiles` or `children`

- *METHOD* is either `Replace` or `Update`
- *DATE_FORMAT* is a date format text string; for example, MMM d, yyyy

Example

```
epmautomate importProfiles NewRecProfiles.csv Profiles Replace "MMM d, yyyy"
```

importRates

Imports currency rates from a CSV file in the Account Reconciliation repository.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importRates` *PERIOD RATE_TYPE REPLACE_MODE FILE_NAME* where:

- *PERIOD* is the name of a period
- *RATE_TYPE* is a predefined rate type
- *REPLACEMENT_MODE* is `Replace` or `ReplaceAll`
- *FILE_NAME* is the name of the CSV file from which rates are to be imported

Example

```
epmautomate importRates "January 2015" Actual ReplaceAll avgrates.csv
```

importRCAttributeValues

Imports attribute values into Account Reconciliation Reconciliation Compliance list or group attributes.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User
Power Users may require additional security provided through ACLs.

Usage

`epmautomate importRCAttributeValues` *ATTRIBUTE_NAME FILE_NAME* `[METHOD=REPLACE|REPLACE ALL|UPDATE][DATEFORMAT=DD/MM/YYYY|DD-MMM-YYYY|MMM d,yyyy|All]`, where:

- *ATTRIBUTE_NAME* is the name of a list or group attribute into which the values are to be imported.

- *FILE_NAME* is a CSV import file from which the values are to be imported. Use the uploadFile command to upload this file to the environment before running this command.
- METHOD, optionally, is how the values are to be imported. Valid values:
  - Replace to add all values from the import file as the attribute value in Reconciliation Compliance. Existing attribute values will be replaced with those in the import file; values not existing in the attribute, but are present in the import file, will be added. Values existing in the attribute that are not in the import file will not be changed. Note that all attribute data for a particular key value will be replaced with the contents from the file or cleared. New values will be added at the bottom in the order they appear in the file.
    This type of import is most useful when you are only moving the latest changes from a source system, for example, when adding new store data from an acquisition to replace only specified attribute values, if present, with the values in the import file. This is the default.
  - Replace All to replace the existing attribute value with the values from the import. Values existing in the attribute, but are not present in the import file, will be deleted. This import type is most useful for mirroring values from a source system with a full update, for example, to complete weekly updates to synchronize with store data from the source system.
  - Update to replace or add all the values in the import file to the attribute. The existing attribute values will be replaced with those in the import file. Values that are in the import file, but are not present in the attribute, will be added. Values that exist in the attribute, but are not present in the import file, will not be changed. Only the attribute data for a particular key value will be replaced with the contents from the file; data for attributes not available in the file will not be touched. Any key present in the import file, but not in the attribute, will cause an error.
    This type of import is most useful for updating a few attributes across all attribute values, for example, while updating the store managers after a reorganization without affecting the rest of the store data.
- Dateformat, optionally, specifies the valid date formats (for example, DD/MM/YYYY, DD-MMM-YYYY (default), MMM d,yyyy, and All) to parse. You may specify multiple date format values separated using a semicolon.

Example

```
epmautomate importRCAttributeValues Stores StoreData.csv METHOD=Replace
DATEFORMAT="All"
```

importReconciliationAttributes

Imports reconciliations attributes into existing reconciliations from a file that you uploaded to the Account Reconciliation environment using the uploadFile command.

Applies To

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importReconciliationAttributes FILE.CSV Period [Rules=RULE_NAME] [Reopen=true|false] [Dateformat=DATE_FORMAT]` where:

- `FILE` is the name of the CSV file that contains the reconciliation attributes that you want to import into reconciliations.
- `Period` identifies the period to which the reconciliations belong.
- `Rules`, optionally, identifies the rules that are to be run on affected reconciliations after importing attributes. Use comma to separate multiple rule names. Valid values are:
  - `None`: Runs no rules on affected reconciliations. This is the default value; it must not be combined with other values.
  - `ALL`: Runs all rules defined for the reconciliations for the specified period. This value must be used by itself; it cannot be combined with other rule names.
  - `SET_ATTR_VAL`: Runs the predefined rule to set attribute value.
  - `CRT_ALT`: Runs the predefined rule to create alert.
  - `AUTO_APP`: Runs the predefined rule to automatically approve the reconciliation.
  - `AUTO_SUB`: Runs the predefined rule to automatically submit the reconciliation.
  - `EMAIL_ON_SAVE`: Runs the predefined rule to automatically send email after updating the reconciliation.
- `Reopen`, optionally, specifies whether to reopen changed reconciliations upon completion of the import operation. Default is `false`.
- `Dateformat`, optionally, specifies the valid date formats (for example, `MM-dd-yyyy`, `dd-MMMM-yy`, `MMM d`, and `yyyy` to parse. You may specify multiple date format values separated using a semicolon.

Examples

- **Importing attribute values for a period and running multiple rules with many date formats:**
  ```
  epmAutomate importReconciliationAttributes Reconciliations.csv "July 2020" Rules=SET_ATTR_VAL,CRT_ALT,AUTO_APP,AUTO_SUB" Reopen=true "Dateformat=MM-dd-yyyy;dd-MMM-yy;MMM d, yyyy"
  ```
- **Importing attribute values for a period without running rules:**
  ```
  epmAutomate importReconciliationAttributes Reconciliations.csv "July 2020"
  ```
- **Importing attribute values for a period, running all applicable rules and reopening affected reconciliations:**
  ```
  epmAutomate importReconciliationAttributes Reconciliations.csv "July 2020" Rules=ALL Reopen=true
  ```

importSnapshot

Imports the contents of a snapshot into the service environment. The snapshot you import must be available in the default upload location.

Use the uploadFile command to upload a snapshot or the copySnapshotFromInstance command to copy it from another instance.

- The following are not part of Planning, Planning Modules, and FreeForm application snapshots:
  - Audit data
  - Job Console data

  Use the cloneEnvironment command or the Clone Environment feature if you want to copy audit and Job Console data to the target environment.
  Snapshot import may fail if the Planning business process contains a renamed seeded period member that has been supplanted by a custom period member. For example, you renamed the seeded *YearTotal* Period member to *unused_YearTotal* and then added an alternate type period member with the original seeded member name (*YearTotal* in this example). In this scenario, import of snapshot may fail.
- Snapshots do not contain the Data Management staging table data. To import this data, use the exportDataManagement and importDataManagement commands or the Data Management System Maintenances Scripts interface. You may use the cloneEnvironment command or the Clone Environment feature to create an identical copy of the environment, including the Data Management staging table data.

The activities that you can complete using this command depend on your role.

- Service Administrators can import only application artifacts into an environment.
- You need both Service Administrator and Identity Domain Administrator roles to import application content into the service environment and identity domain artifacts (users and their predefined role assignments) into the identity domain of the environment.
  If a user who is not in the identity domain is referenced in the snapshot being imported, EPM Automate creates a user in the identity domain and assigns the default password that you specify in the command or a temporary unique password to each user if you do not specify a password in the command. By default, the user will be required to reset password during first sign in.

> **Note:**
>
> - For business processes other than Account Reconciliation, Profitability and Cost Management, and Oracle Enterprise Data Management Cloud: While loading metadata, Oracle Enterprise Performance Management Cloud may make multiple loading passes if the previous attempt resulted in rejected records because shared members come before base members in the outline. Such attempts may increase the command processing time.
> - Users who are members of groups in Access Control must be assigned to a predefined role. Attempts to assign a user, who is not assigned to a predefined role, to a group is not permitted.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Identity Domain Administrator role is required to import users and predefined role assignments.

Usage

`epmautomate importSnapshot SNAPSHOT_NAME [importUsers=true|false] [userPassword=DEFAULT_PASSWORD] [resetPassword=true|false]` where:

- *SNAPSHOT_NAME* is the name of a snapshot in the default upload location.
- `importUsers`, optionally, specifies whether to import users and their predefined role assignments from the snapshot. Default is `false`. Use `importUsers=true` to import users and predefined role assignments into the identity domain if the source snapshot contains data on new users or if new roles have been assigned to current users.
  User login values are not case-sensitive. For example, the user login value `jane.doe@example.com` is treated as being identical to `Jane.Doe@Example.com` or any variation in its case. If any variation matches a user login existing in the identity domain, this command does not import the user from the snapshot.

  > **Note:**
  >
  > – Import of users and their predefined roles fails if a user who is not an Identity Domain Administrator performs the import operation. The following error is recorded in the Migration Status Report: `Failed to import External Directory Artifact ARTIFACT_NAME. User USER_NAME is not authorized to perform this operation. The user needs to have Identity Domain Administrator role to perform this operation.`
  >
  > – If you are not importing users and a user in the source snapshot is not assigned to a predefined role on the target environment, an error (`EPMIE-00070: Failed to find user during assigned roles import`) is displayed.

  – Changes to the predefined roles of the user will be updated based on the roles assigned in the source snapshot. However, role assignments in the target will not be removed to match those in the source snapshot. For example, assume that `jdoe` is assigned to the Power User predefined role in the target environment, but has only the User role in the source snapshot. In this situation, this command assigns `jdoe` to the User role and does not remove the Power User role assignment in the target environment.

  – This command does not delete existing users from the target environment if they don't exist in the source snapshot. For example, `jdoe` has an account in the target environment, but this account is not present in the source snapshot. In this situation, the account of `jdoe` in the target environment is not deleted.

  – This command adds users that do not exist in the target environment; it does not update current user properties in the target environment even if those are different in the source snapshot. For example, if the last name of `jdoe` in the source snapshot is spelled differently in the target environment, the change will not be made in the target environment.

  – This command does not change existing users' passwords in the target environment even if it is different in the source snapshot.

- `userPassword`, optionally, indicates the default password to assign to new users who are created in the identity domain. The password that you specify must meet the minimum password requirements. If you do not specify a value for this parameter, a unique temporary password is assigned to each user.
- *`resetPassword`*, optionally, indicates whether the new user must change the password at the first log in. Default is `true`, requiring new users to change the password at the first sign in. If this value is set to `true`, new users will receive account activation emails prompting them to change passwords.

Examples

- Import application artifacts only: `epmautomate importSnapshot April16FullApp`
- Import application and identity domain artifacts (requires Service Administrator and Identity Domain Administrator roles):
  - Assign a unique temporary password to each new user and force them to reset their password after they sign in for the first time:
    ```
    epmautomate importSnapshot April16FullApp importUsers=true
    ```
  - Assign a specific password and allow users to not change it if they so choose. Not recommended for imports into production environments:
    ```
    epmautomate importSnapshot April16FullApp importUsers=true
    userPassword=P@ssw0rd1 resetPassword=false
    ```

importSupplementalCollectionData

Imports supplemental collection data from a file into the application.

Use the uploadFile command to upload the file containing the data to the default upload location. the import file format is as follows:

```
#Workflow
Workflow_Dimension_1_Name,Workflow_Dimension_2_Name,Workflow_Dimension_n_Name
Workflow_Dimension_1_Member,Workflow_Dimension_2_Member,Workflow_Dimension_n_M
ember
#Collection
Collection_Attribute_1,Collection_Attribute_2,Collection_Attribute_n
Record1_Attr_Value_1,Record1_Attr_Value_2, Record1_Attr_Value_n
```

For example:

```
#Workflow
Entity
9100
#Collection
Custody Account Code,Trade Currency Code,Account Description,Base Currency
Code,CIC Code,IFRS 13 Tier,SII Portfolio Type,WPM Detailed NAV ID,WPM Asset
Description
1,,,,111,,,,6
```

Applies to

Financial Consolidation and Close, and Tax Reporting.

Required Roles

Service Administrator

Usage

> **Note:**
>
> All command parameters must be enclosed in double quotation marks.

`epmautomate importSupplementalCollectionData "FILE_NAME""COLLECTION_NAME" "YEAR" "PERIOD" "[FREQUENCY_DIMENSION=MEMBER]"` where:

- `FILE_NAME` is the name of a CSV file, available in the default upload location, that contains properly formatted supplemental data.
- `COLLECTION_NAME` is the name of the collection into which the supplemental data in the file should be imported.
- `YEAR` is the year dimension member to be used for collection.
- `PERIOD` is name of the period dimension to be used for collection.
- *`FREQUENCY_DIMENSION`*, optionally, is the name of the frequency dimension to be used for collection. You may specify as many frequency dimensions as needed in *`"FREQUENCY_DIMENSION1=MEMBER" "FREQUENCY_DIMENSION2=MEMBER"`* format.

Example

```
epmautomate importSupplementalCollectionData "datafile.csv" "Journal Data
Collection" "FY20" "Jan" "Account=PAYROLL" "JournalID=LNR 113"
```

importSupplementalData

Imports supplemental data from a file into the application.

Use the uploadFile command to upload the file containing the data to the default upload location.

Applies to

Financial Consolidation and Close and Tax Reporting.

Required Roles

Service Administrator

Usage

> **Note:**
>
> All command parameters must be enclosed in double quotation marks.

`epmautomate importSupplementalData "FILE_NAME" "DATA_SET_NAME" "YEAR" "PERIOD_NAME" "SCENARIO_NAME"` where:

- `FILE_NAME` is the name of a CSV file, available in the default upload location, that contains properly formatted supplemental data.
- `DATA_SET_NAME` is the name of the data set into which the supplemental data in the file should be imported.
- `YEAR` is the year for which the data set is deployed.
- `PERIOD_NAME` is name of the period to which the data set is deployed.
- `SCENARIO_NAME` is the name of the scenario to which the data set is deployed.

Example

```
epmautomate importSupplementalData "DatasetImport.csv" "EmployeeDataSet" "FY17"
"Jan" "Actual"
```

importTemplate

Creates an application structure by importing from a template file that exists in profitinbox.

You can upload a template file into profitinbox using the uploadFile command.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate importTemplate` *`APPLICATION_NAME File_Name`* `isApplicationOverwrite=true|false` where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that you want to create by importing the template
- `File_Name` is the name of the .ZIP file containing application template. This file must exist in profitinbox.
- `isApplicationOverwrite` specifies whether to overwrite the existing application, if any. Specify this parameter value in all lower case.

Example

```
epmautomate importTemplate BksML12 template1.zip isApplicationOverwrite=true
```

importTMAttributeValues

Imports values into Account Reconciliation Transaction Matching group attributes.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User
Power Users may require additional security provided through ACLs.

Usage

`epmautomate importTMAttributeValues ATTRIBUTE_NAME FILE_NAME [METHOD=REPLACE|REPLACE ALL|UPDATE][DATEFORMAT=DD/MM/YYYY|DD-MMM-YYYY|MMM d,yyyy|All]`, where:

- *ATTRIBUTE_NAME* is the name of a group attribute into which the values are to be imported.
- *FILE_NAME* is a CSV import file from which the values are to be imported into Transaction Matching. Use the uploadFile command to upload this file to the environment before running this command.
- `METHOD`, optionally, is how the values are to be imported. Valid values:
  - `Replace` to add all values from the import file into Transaction Matching group attributes. Existing attribute values will be replaced with those in the import file; values not existing in the attribute, but are present in the import file, will be added. Values existing in the attribute that are not in the import file will not be changed. Note that all attribute data for a particular key value will be replaced with the contents from the file or cleared. New values will be added at the bottom in the order they appear in the file. This type of import is most useful when you are only moving the latest changes from a source system, for example, when adding new store data from an acquisition to replace only specified attribute values, if present, with the values in the import file. This is the default.
  - `Replace All` to replace the existing attribute value with the values from the import. Values existing in the attribute, but are not present in the import file, will be deleted. This import type is most useful for mirroring values from a source system with a full update, for example, to complete weekly updates to synchronize with store data from the source system.
  - `Update` to replace or add all the values in the import file to the attribute. The existing attribute values will be replaced with those in the import file. Values that are in the import file, but are not present in the attribute, will be added. Values that exist in the attribute, but are not present in the import file, will not be changed. Only the attribute data for a particular key value will be replaced with the contents from the file; data for attributes not available in the file will not be touched. Any key present in the import file, but not in the attribute, will cause an error.
    This type if import is most useful for updating a few attributes across all attribute values, for example, while updating the store managers after a reorganization without affecting the rest of the store data.
- `Dateformat`, optionally, specifies the valid date formats (for example, `DD/MM/YYYY`, `DD-MMM-YYYY` (default), `MMM d,yyyy`, and `All`) to parse. You may specify multiple date format values separated using a semicolon.

Example

```
epmautomate importTMAttributeValues TMGA TMGA.csv METHOD=Replace DATEFORMAT="All"
```

importTmPremappedTransactions

For a specific data source, imports pre-mapped transactions data from a file in Account Reconciliation repository into Transaction Matching.

Use the uploadFile command to upload the transactions file to the service.

This command displays import status and an import log file name in the console. Use the downloadFile command to download the log file to a local computer.

See Importing Data in *Reconciling Accounts with Account Reconciliation* for import file format requirements and information about importing data.

> **Note:**
>
> - You can import transactions for only one match type at a time. However, parallel imports can be run into different match types.
> - Unlike from the Jobs screen, you can import pre-mapped transactions data only from one file at a time.
> - After importing pre-mapped transactions for all data sources, run the `runautomatch` command.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate importTmPremappedTransactions` *`MATCH_TYPE DATA_SOURCE FILE_NAME [DATE_FORMAT]`* where:

- `MATCH_TYPE` is a match type defined in Account Reconciliation.
- `DATA_SOURCE` is the identifier of the data source associated with the reconciliation type that you specified.
- `FILE_NAME` is the name of the CSV file containing the transactions to import. This file must be available in the service.
- `DATE_FORMAT` is an optional parameter that indicates the format of the date fields included in the transactions import file. Default is `dd-MMM-YYYY`. Other supported date formats are: `MM/dd/yyyy`, `dd/MM/yyyy`, `MM-dd-yyyy`, `d-M-yyyy`, and `MMM d.yyyy`.

Example

```
epmautomate importTmPremappedTransactions "INTERCOMPANY" "AP" dailydata.csv d-M-yyyy
```

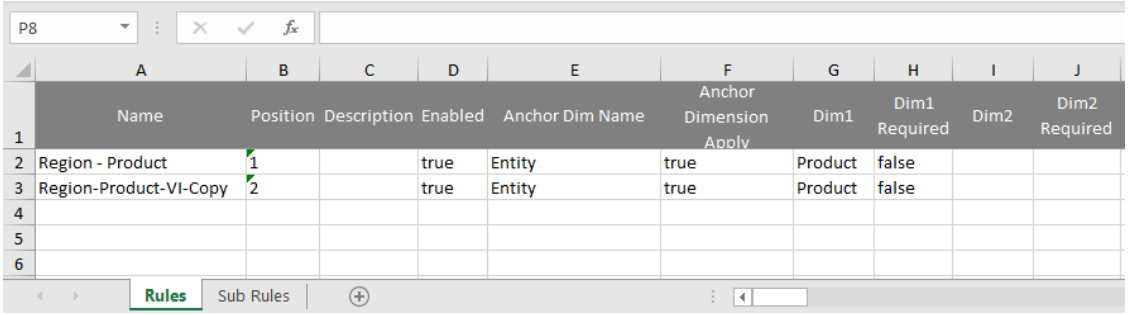
importValidIntersections

Imports valid intersection groups from a ZIP file that contains one Excel file with valid intersection definitions into the business process. Before running this command, use the uploadFile command to upload the import file to the environment.

Your import ZIP file should contain an Excel file with two worksheets (Rules and Sub Rules) for successfully importing valid intersections. The first sheet, Rules, should define the intersection group, dimensions included, and properties such as Unspecified Valid, Additional Dims Required. The second sheet, Sub Rules, should provide member selections and exclusions. For more information, see these topics in *Administering Planning*.

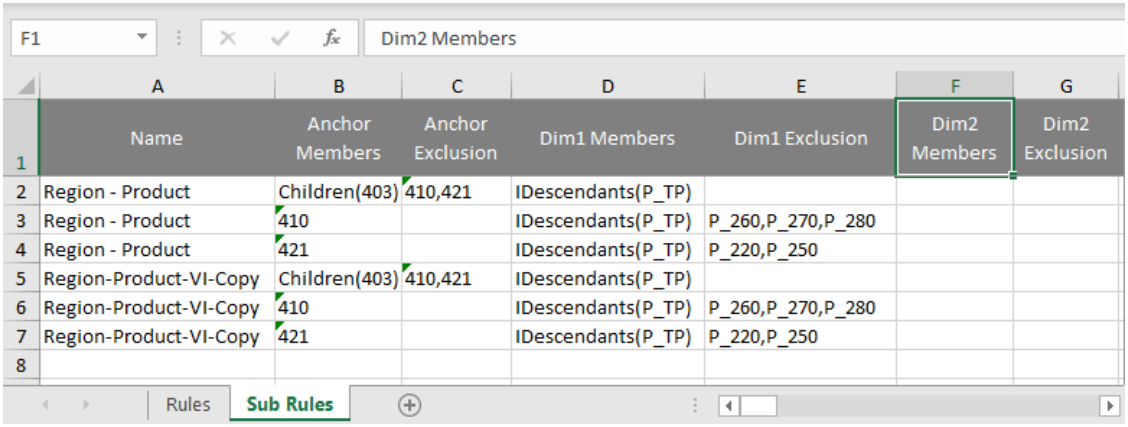
- Anchor and Nonanchor Dimensions
- Valid Intersection Examples

The best method to get the import file format template is to export valid intersections from the application. A sample format is presented in the following illustrations.



	A	B	C	D	E	F	G	H	I	J
1	Name	Position	Description	Enabled	Anchor Dim Name	Anchor Dimension Apply	Dim1	Dim1 Required	Dim2	Dim2 Required
2	Region - Product	1		true	Entity	true	Product	false		
3	Region-Product-VI-Copy	2		true	Entity	true	Product	false		
4										
5										
6										

Rules | Sub Rules



	A	B	C	D	E	F	G
1	Name	Anchor Members	Anchor Exclusion	Dim1 Members	Dim1 Exclusion	Dim2 Members	Dim2 Exclusion
2	Region - Product	Children(403)	410,421	IDescendants(P_TP)			
3	Region - Product	410		IDescendants(P_TP)	P_260,P_270,P_280		
4	Region - Product	421		IDescendants(P_TP)	P_220,P_250		
5	Region-Product-VI-Copy	Children(403)	410,421	IDescendants(P_TP)			
6	Region-Product-VI-Copy	410		IDescendants(P_TP)	P_260,P_270,P_280		
7	Region-Product-VI-Copy	421		IDescendants(P_TP)	P_220,P_250		
8							

Rules | Sub Rules

Applies To

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate importValidIntersections` *`FILE_NAME`*`.zip [ErrorFile=`*`ERROR_FILE_NAME`*`.txt]` where:

- `FILE_NAME` is the name of the ZIP file that contains the valid intersection definition Excel file.
- `ErrorFile`, optionally, identifies the name of the text file to which error records will be written. If this parameter value is not specified, EPM Automate automatically generates an error file; you can view its name in the Job Console.

Example

```
epmautomate importValidIntersections VI_Import_File.zip
ErrorFile=VI_Import_Log.txt
```

invalidLoginReport

In OCI (Gen 2) environments, creates the Invalid Login Report, which lists the failed attempts to sign into the environment over a specified period of time corresponding to the audit retention period specified for your environment. The default retention period is 30 days. You can extend it to a maximum of 90 days by changing the **Audit Retention Period (days)** setting in the Oracle Cloud Identity Console. To retain the audit data for duration longer than 90 days, periodically download and archive this report and the Role Assignment Audit Report.

The Invalid Login Report contains information such as the following:

- User name of the user who attempted to sign in
- Remote IP address from which the user attempted to sign in
- Timestamp of the sign in attempt

This report shows all the unsuccessful login attempts to the corresponding Identity Cloud Service. These may not all be related to one Oracle Enterprise Performance Management Cloud instance.

	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

Use the downloadFile command to download the report to a local computer.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

Usage

`epmAutomate invalidLoginReport` *`FROM_DATE TO_DATE FILE_NAME`*`.CSV` where:

- *`FROM_DATE`* indicates the start date (in `YYYY-MM-DD` format) of the period for which the report is to be generated. This date must fall within the audit data retention period specified in the Oracle Cloud Identity Console.
- *`TO_DATE`* indicates the end date (in `YYYY-MM-DD` format) of the period for which the report is to be generated.

- *FILE_NAME* is the name of a CSV file for the report.

> **Note:**
>
> This report can be generated only for the last 90 days.

Example

```
epmAutomate invalidLoginReport 2021-06-01 2021-06-30 invalidLoginReport.CSV
```

listBackups

Lists available backup snapshots of OCI (Gen 2) environments to determine if a specific backup is available so that you can archive it or use it to restore the current environment by yourself. This command does not work in Classic Oracle Enterprise Performance Management Cloud environments.

Before trying to restore a specific backup, use this command to check if the required backup is available in Oracle Object Storage. If the backup is available, you can restore it (copy it to your environment) by running the restoreBackup command. After copying the backup, you can import it using importSnapshot command. Self-service restoration of the environment saves you processing time.

For services other than Narrative Reporting, this command lists available backup snapshots (created by the daily maintenance process) using the naming convention `YYYY-MM-DDTHH:MM:SS/Artifact_Snapshot.zip`; for example, `2022-02-16T21:00:02/Artifact_Snapshot.zip`. For Narrative Reporting, available snapshots use the naming convention `YYYY-MM-DDTHH:MM:SS/EPRCS_Backup.tar.gz`; for example, `2022-02-16T21:00:02/EPRCS_Backup.tar.gz`. In both cases, the timestamp reflects the UTC time when the snapshot was created. The following illustration displays a sample command output.

```
c:\Oracle\EPM Automate\bin>epmautomate listbackups

2022-03-04T06:37:51/Artifact_Snapshot.zip
2022-03-08T06:32:01/Artifact_Snapshot.zip
2022-03-09T12:08:05/Artifact_Snapshot.zip
2022-03-10T06:37:48/Artifact_Snapshot.zip
2022-03-15T06:21:28/Artifact_Snapshot.zip
2022-03-16T06:20:52/Artifact_Snapshot.zip
2022-03-16T12:13:56/Artifact Snapshot.zip

Total 7
```

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

```
epmAutomate listBackups
```

Example

```
epmAutomate listBackups
```

listFiles

Lists the names of the files in the default location, Data Management folders, and profitinbox/profitoutbox (Profitability and Cost Management).

This command also lists incremental and backup export files, Migration snapshots, access logs, and Activity Reports. This illustration shows a truncated version of the command output.

```
apr/2022-01-27 05_23_36/activityreport.json
apr/2022-01-28 05_24_07/2022-01-28 05_24_07.html
apr/2022-01-28 05_24_07/access_log.zip
apr/2022-01-28 05_24_07/activityreport.json
apr/2022-01-29 05_24_06/2022-01-29 05_24_06.html
apr/2022-01-29 05_24_06/access_log.zip
outbox/Vision_99.dat
roleassign.csv
RoleAssignment.csv
sanity_no_data_22-01-18.zip
U-1.csv
U2.csv
user1.csv
user12.csv
users12.csv
Uservariables-MemberFormula.zip
UsrGrpReport.CSV
Vision_DTsetup.zip
VisionADCForms2010.zip
```

This command will not list the current snapshot if this command is executed while the snapshot of the environment is being generated; for example, during the daily maintenance.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

```
epmautomate listFiles
```

Example

```
epmautomate listFiles
```

loadData

Loads data into a calculation cube using a file available in profitinbox.

Use the uploadFile command to load files into profitinbox.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate loadData` *`APPLICATION_NAME`* `dataFileName=`*`File_Name`* `PARAMETER=`*`VALUE`*
where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application into which you want to load data
- `dataFileName=`*`File_Name`* specifies a data load file available in profitinbox.
  Data file name must be enclosed in double quotation marks.
- `PARAMETER=`*`VALUE`* indicates runtime parameters and their values to load data. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:
  - `clearAllDataFlag=true|false` specifies whether to clear existing data in the application cube
  - `dataLoadValue=OVERWRITE_EXISTING_VALUES|ADD_TO_EXISTING` specifies how to handle existing data

Example

```
epmautomate loadData BksML12 dataFileName="data1.txt"clearAllDataFlag=true
dataLoadValue="OVERWRITE_EXISTING_VALUES"
```

loadDimData

Loads dimension metadata from one or more files in profitinbox into an application.

Use the uploadFile command to load metadata files into profitinbox.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate loadDimData APPLICATION_NAME dataFileName=File_Name [stringDelimiter="DELIMITER"] [acceptableDecreasePercentage=PERCENTAGE]` where:

- *APPLICATION_NAME* is the name of the Profitability and Cost Management application into which you want to load dimension metadata
- `dataFileName` specifies a dimension metadata load file available in profitinbox. To load metadata from multiple files, list the file names separated by a delimiter
- `stringDelimiter`, optionally, specifies the delimiter used to separate metadata file names. Delimiter must be enclosed in double quotation marks.
- `acceptableDecreasePercentage`, optionally, specifies the percentage (without the % symbol) difference in member count that will be acceptable for the operation. If the new member count from the incoming file is less than the existing member count, this value represents the percentage decrease that is acceptable. The loading of dimension data fails if the deviation of member count exceeds this percentage.

Example

```
epmautomate loadDimData BksML12 dataFileName="dimdata1.txt#dimdata1.txt"
stringDelimiter="#" acceptableDecreasePercentage=5
```

loadViewpoint

Loads a viewpoint (a subset of nodes) from a load file into an Oracle Enterprise Data Management Cloud application.

Viewpoint loads enable you to load data into viewpoints that are unbound, bound, or partially bound. The viewpoint load file, a CSV, Excel (XLSX) file or a ZIP file containing one CSV or XLSX file, must be available in the environment where you are loading the viewpoint. You can upload the load file to the environment using the uploadFile or copyFileFromInstance command.

Applies to

Oracle Enterprise Data Management Cloud

Required Roles

Service Administrator

Usage

`epmautomate loadViewpoint VIEW VIEWPOINT PURPOSE FILE_NAME [loadType=ReplaceNodes|Merge]`
, where:

- `VIEW` is the name of an Oracle Enterprise Data Management Cloud view.
- `VIEWPOINT` is the name of the viewpoint that you want to load.
- `PURPOSE` is a text string, enclosed in double quotation marks, indicating why the viewpoint is being loaded.
- `FILE_NAME` is the name of the file, with extension, from which the viewpoint is to be loaded.

- `loadType`, optionally, identifies how to load viewpoint. Valid values are `Merge` and `ReplaceNodes`.
  - Use `Merge` to preserve existing relationships by processing incremental changes.
  - Use `ReplaceNodes` to clear all relationships (including orphan relationships and relationships used by other viewpoints using the same hierarchy set) from the hierarchy other than those from the load file. This is the default load type.

Examples

- **Merge incremental changes:** `epmautomate loadViewpoint USOperations Entity "Daily Upstream Load" data_Entity.CSV loadType=Merge`
- **Replace existing hierarchies:** `epmautomate loadViewpoint USOperations Entity "Replace US Operations data" data_Entity.CSV`

login

Establishes a secure connection to an environment. This command supports signing into an environment using a plain text password, or an encrypted password file containing the password or OAuth 2.0 refresh token. Login using OAuth 2.0 refresh token is supported for OCI (Gen 2) environments only.

You sign in to initiate a session, which remains active until you sign out.

> **Note:**
>
> - This command is not supported for users who are set up for basic authentication with multi-factor authentication (MFA).
> - EPM Automate does not support signing in with your organization's SSO credentials.
> - EPM Automate does not work with SOCKS proxy; it works only with HTTP/HTTPS proxy.
> - When using this command in batch files to automate activities, Oracle recommends that you use encrypted password or OAuth 2.0 refresh token to avoid recording clear text passwords in batch files.
> - On Windows computers, this command automatically identifies missing proxy server intermediate security certificate that may prevent you from establishing a connection and adds it to the JRE installed under `C:\Oracle\EPM Automate`. This prevents login errors related to security certificates when using proxy servers to access the internet.
>
> On Linux computers, the login command identifies the missing security certificate from the proxy server, downloads it, and display an error. A user with `root` access can then install the downloaded certificate in the JRE available in the `JAVA_HOME` identified in the environment variables. See these information sources:
>
> - Java Runtime Environment and EPM Automate
> - Keytool Java documentation

On signing in, a message to upgrade EPM Automate is displayed if you are using an older version. You can use the upgrade command to silently upgrade your installation.

If you plan to run addUsers, removeUsers, assignRole, or unassignRole command, do not login using the OAuth refresh token. These commands require you to use basic authentication. All other commands work with OAuth 2.0 in OCI (Gen 2) environments.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

- **Using unencrypted password:** `epmautomate login USERNAME PASSWORD URL [IDENTITYDOMAIN] [ProxyServerUserName=PROXY_USERNAME ProxyServerPassword=PROXY_PASSWORD ProxyServerDomain=PROXY_DOMAIN] ] [KeystorePassword=PASSWORD]`
- **Using encrypted file:** `epmautomate login USERNAME PASSWORD_FILE URL [IDENTITYDOMAIN] [ProxyServerUserName=PROXY_USERNAME] [ProxyServerPassword=PROXY_PASSWORD] [ProxyServerDomain=PROXY_DOMAIN] [KeystorePassword=KEYSTORE_PASSWORD]`

In these commands:

- *USERNAME* is the user name of the user.
- *PASSWORD* is the password of the user.
- *PASSWORD_FILE* is the name and location of the file that stores the encrypted password or OAuth 2.0 refresh token of the user. See the encrypt command.
- *URL* is the base URL of the environment to which to connect. You may use a custom or vanity URL in place of the Oracle Enterprise Performance Management Cloud URL. See Using Vanity URLs in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*

> **Note:**
>
> If using an API gateway or reverse proxy, use its URL and the context defined for your environment in place of the EPM Cloud URL.

- *IDENTITYDOMAIN*, optionally, is the identity domain of the environment.
  This value is derived automatically from the EPM Cloud URL; any value you specify is ignored. However, this value is required if you are using an API gateway or reverse proxy URL to connect to a Classic EPM Cloud environment.
- `ProxyServerUserName` is the user name to authenticate a secure session with the HTTP proxy server that controls access to the internet. Specify the user name without prefixing a domain name prefix. Required only if authentication at proxy server is enabled for your network.

- `ProxyServerPassword` is the password to authenticate the user with the proxy server. Required only if authentication at proxy server is enabled for your network. This password can be encrypted. See the encrypt command. If this password is encrypted, it is read from the *`PASSWORD_FILE`*.
- `ProxyServerDomain` is the name of the domain defined for the HTTP proxy server (not the server name or the proxy server host name). Required only if authentication at proxy server is enabled for your network and a proxy server domain is configured.
- `KeystorePassword`, optionally, is the keystore password required for importing proxy server security certificate. Use this parameter only on Windows, and only if you are faced with the following errors in environments where a proxy server is being used to channel internet access:

```
EPMAT-7: Unable to connect as few SSL certificates are missing in the keystore

EPMAT-7: Unable to connect as above-mentioned SSL certificates are missing in the
keystore
```

> **Note:**
>
> EPM Automate detects and uses the HTTP/HTTPS proxy settings on your computer. EPM Automate supports the following authentication mechanisms to connect to the proxy server:
>
> - Basic authentication
> - Digest authentication
> - Kerberos authentication
> - Negotiate proxy authentication
> - NTLM authentication
>
> The available authentication method and its configuration depends on the proxy server you are using.
>
> On Linux computers, if the proxy settings require you to authenticate with the proxy server, you must enter the proxy server domain, user name, and password as parameters to this command. Contact your network administrator for help with proxy server domain name and credentials.

Examples

- Using an unencrypted EPM Cloud password, no proxy authentication:
  ```
  epmautomate login serviceAdmin P@ssword1 https://test-cloud-
  pln.pbcs.us1.oraclecloud.com
  ```
- Using an encrypted file, no proxy authentication:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-
  pln.pbcs.us1.oraclecloud.com
  ```
- Using an encrypted file, if authentication at proxy server is enabled with server domain:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-
  pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com
  ProxyServerPassword=example ProxyServerDomain=example
  ```
- Using an encrypted file, if authentication at proxy server is enabled without a server domain:

```
epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com ProxyServerPassword=example
```

- Using encrypted EPM Cloud and proxy server password, if authentication at proxy server is enabled with a server domain:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com ProxyServerDomain=example
  ```
- Using encrypted EPM Cloud and proxy server password, if authentication at proxy server is enabled without a server domain:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com
  ```
- Using an encrypted file with APIGEE API gateway:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://exampleapigee.apigee.com/epm example_ID_DOM
  ```
- Using a vanity URL:
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://rebrand.ly/Automate
  ```

logout

Terminates your current connection with an environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

```
epmautomate logout
```

Example

```
epmautomate logout
```

maskData

Masks application data to ensure data privacy. Use this command only on test environments to hide sensitive data from application developers.

WARNING: Do not use this command on production environments because it randomizes current application data, rendering it meaningless. You cannot undo the effects of this command. If you mistakenly masked the data in a service environment, you must restore the data from a backup or from the maintenance snapshot.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate maskData [-f]` where `-f` is an option to force the start of the masking process without user confirmation. If you do not use the `-f` option, EPM Automate prompts you to confirm your action.

Example

```
epmautomate maskData [-f]
```

mergeDataSlices

Merges all incremental data slices of an aggregate storage cube into the main database slice and, optionally, removes cells that have a value of zero.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate mergeDataSlices` *`CUBE_NAME`* `[keepZeroCells=true|false]` where:

- `CUBE_NAME` identifies the aggregate storage cube for which all data slices are to be merged.
- `keepZeroCells`,optionally, specifies whether to remove cells that have a value of zero (logically clearing data from a region results in cell with a value of zero). Default is `true`

Example

```
epmautomate mergeDataSlices rep1 keepZeroCells=false
```

mergeSlices

Merges incremental data slices into the main database cube and, optionally, removes the Oracle Essbase cells containing 0 (zero) as value to make the cube compact.

Removing cells containing 0 optimizes cube performance.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate mergeSlices applicationName [removeZeroCells=true|false]` where:

- `applicationName` is the name of an Profitability and Cost Management application.
- `removeZeroCells`, optionally, specifies whether to remove cells containing 0. Default value of this parameter is `false`.

Examples

- Merge slices without removing cells containing 0s:
  - `epmautomate mergeSlices BksML30`
  - `epmautomate mergeSlices BksML30 removeZeroCells=false`
- Merge slices and remove cells containing 0s: `epmautomate mergeSlices BksML30 removeZeroCells=true`

optimizeASOCube

Optimizes the performance of queries for selecting aggregate views for data extraction from ASO cubes.

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. 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, or Data Management, and any other MDX queries representative of the type of queries for which optimization is desired to train Oracle Essbase.
- Create aggregation based on optimized or default queries.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate optimizeASOCube` *`APPLICATION_NAME OPTIMIZATION_TYPE`* where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application to which the ASO cube belongs.
- *`OPTIMIZATION_TYPE`* is a cube optimization operation. Acceptable values are:
  - `clearAggregations` which removes default and query-based views.

  - `createAggregations` which creates default Essbase aggregate views. Use this option to perform default aggregation instead of query-based aggregation
  - `startQueryTracking` which starts query tracking.
  - `stopQueryTracking` which 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. Essbase can aggregate views based on data collected until query tracking is stopped.
  - `createQBOAggregations` which creates Essbase aggregate views based on the optimized queries that you run after enabling query tracking.

Examples

- Drop default and query-based aggregate views:
  `epmautomate optimizeASOCube BksML12 clearAggregations`
- Start query tracking
  `epmautomate optimizeASOCube BksML12 startQueryTracking`
- Create Essbase aggregate views based on the optimized queries that you run after starting query tracking:
  `epmautomate optimizeASOCube BksML12 createQBOAggregations`

programDocumentationReport

Creates the Program Documentation Report containing Profitability and Cost Management application logic.

You can download the report to a local computer using the downloadFile command.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate programDocumentationReport` *`APPLICATION_NAME POV_NAME`* `[fileName=`*`FILE_NAME`*`] [fileType=PDF|WORD|EXCEL|HTML] [useAlias=true|false] stringDelimiter="`*`DELIMITER`*`"` where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application for which the Program Documentation Report is to be created.
- *`POV_NAME`* is the name of the model POV in the application for which the report is to be generated.
- `fileName`, optionally, is a unique name (including extension) for the report file. Default report file name is `HPCMMLProgramDocumentationReport_`*`APPLICATION_NAME_POV_NAME`*`.pdf`.
- `fileType`, optionally, is the output file format. Default is `PDF`.
- `useAlias`, optionally, specifies whether to print aliases in place of member names. Default is `false`.

- `stringDelimiter` is the delimiter used in POV values. Delimiter must be enclosed in double quotation marks.

Example

```
epmautomate programDocumentationReport BksML30 2019_Feb_Actual fileName=Feb-
Actual.xls fileType=Excel useAlias=true stringDelimiter="_"
```

provisionReport

Generates a Role Assignment Report (.CSV file) and stores it in the default download location.

The report lists the predefined roles (for example, Service-name Power User) and application roles (for example, Mass Allocation, which is a Planning application role) assigned to users. Use the downloadFile command to download the report.

Two versions of the report can be generated: simplified or classic. The simplified report, which is identical to the Role Assignment Report that is available from the Access Control screen, does not list the application roles that are subsumed into predefined roles or the component roles of application roles assigned to the user. The classic version of the report lists the component roles that are subsumed into the predefined roles to which users are assigned. It also lists the application roles assigned to the user (directly or through groups).

Generating this report refreshes the user and role information available in Access Control.

For OCI (Gen 2) only: Oracle Enterprise Performance Management Cloud considers deactivated users as being identical to users not assigned to any predefined roles even though such users may have had predefined roles when they were deactivated. Information on deactivated users is not included in this report.

> **Note:**
>
> This command will be deprecated in an upcoming release. Instead of this command, use the roleAssignmentReport command, which produces an equivalent report.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate provisionReport` *`REPORT_NAME`* `[format=classic|simplified] [userType=serviceUsers|IDAdmins]` where:

- *`REPORT_NAME`* is a name for the report.
- `format`, optionally, identifies how the report is to be formatted. Acceptable values:
  - `simplified`, the default option, creates a report that is identical to the Role Assignment Report generated from the Access Control screen.

  - `classic` creates a report that lists the component roles that are subsumed into the predefined roles to which users are assigned. It also lists the application roles assigned to the user (directly or through groups)
- `userType`, optionally, identifies the users to be included in the report. If you do not specify a value for this parameter, the default value `serviceUsers` is used. Acceptable values:
  - `serviceUsers` creates a report that contains information on all functional users (does not include Identity Domain Administrators if they are not assigned to a predefined role that grants access to the application)
  - `IDAdmins` creates a report that lists only the users assigned to the Identity Domain Administrator role. The report is identical in classic and in simplified format

Examples

- Create a classic report: `epmautomate provisionReport myProvReport.CSV format=classic`
- Create a simplified report:
  - `epmautomate provisionReport myProvReport.CSV format=simplified`
  - `epmautomate provisionReport myProvReport.CSV userType=serviceUsers`
- Create a report listing only Identity Domain Administrators:
  - `epmautomate provisionReport myProvReport.CSV userType=IDAdmins`
  - `epmautomate provisionReport myProvReport.CSV userType=IDAdmins format=classic`

purgeArchivedTmTransactions

Purges archived matched transactions from the Account Reconciliation application.

You use the archiveTmTransactions command periodically to archive old matched transactions and then run this command to remove them from the Account Reconciliation to ensure optimal application size.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate purgeArchivedTMTransactions JobID=`*`JOB_ID`* where `JobID` is the identifier of the Archive TM Transaction job that was run to archive matched transactions. This job ID is displayed in the EPM Automate console when you execute the archiveTmTransactions command. You can also find it in the Job Console.

Example

```
epmautomate purgeArchivedTMTransactions JobID=100000002655003
```

purgeTmTransactions

Removes matched transactions from Account Reconciliation.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate purgeTmTransactions matchType age [filterOperator=`*`VALUE`*`] [filterValue=`*`VALUE`*`] [logFilename=`*`FILE_NAME`*`]` where:

- `matchType` is the identifier (TextID) of the match type from which matched transactions should be deleted.
- `age` identifies the number of days since the transaction was matched. Matched transaction older than or equal to this value will be deleted.
- `filterOperator`, optionally, is 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:
  - `equals`
  - `not_equals`
  - `starts_with`
  - `ends_with`
  - `contains`
  - `not_contains`
- `filterValue`, optionally, is a filter value to identify the transactions to purge. If the `filterOperator` is `equals` or `not_equals`, you can use a space-separated list to specify multiple values; for example, `filterValue=101-120 filterValue=102-202`. If multiple values are specified, transactions from accounts matching any filter operator and filter value combination are selected for purging.
- `logFilename`, optionally, is the name of a log file to record information about the command activity. If a file name is not specified, a log file named `PurgeTransactions_`*`JOB_ID`* is automatically generated.

> **Note:**
>
> If `filterOperator` and `filterValue` are not specified, all matched transactions older than or equal to the `age` from all accounts for the specified `matchType` are purged.

Examples

- Purge matched transactions `180` days or older for match type `cashrecon`:
  `epmautomate purgeTMTransactions cashrecon 180 logFileName=tmlogs.log`
- Purge matched transactions `180` days or older for match type `cashrecon` for Account `101-120` or `102-202`:
  `epmautomate purgeTMTransactions cashrecon 180 filterOperator=equals filterValue=101-120 FilterValue=102-202`
- Purge matched transactions `180` days or older for match type `cashrecon` for any account containing the string `11`:
  `epmautomate purgeTMTransactions cashrecon 180 filterOperator=contains filterValue=11`

recomputeOwnershipData

Recomputes ownership data

Recomputing of ownership data in Financial Consolidation and Close is required in these situations:

- After adding or deleting override rules for Ownership Management accounts
- After you change Consolidation Methods range settings
- After a database refresh, regardless of whether the entity structure was changed

Recomputing of ownership data in Tax Reporting is required after each database refresh even if the entity structure was not changed.

Applies to

Financial Consolidation and Close and Tax Reporting.

Required Roles

Service Administrator, Power User, User

Usage

`epmautomate recomputeOwnershipData *Scenario Year Period*` where:

- `Scenario` is the name of the scenario to recompute.
- `Year` is the year to recompute.
- `Period` is the first period of the year to recompute.
  The selected period and all subsequent periods are recomputed.

> **Note:**
>
> A POV that requires recomputation can be consolidated only after the ownership data is recomputed.

Example

```
epmautomate recomputeOwnershipData FCCS_total_Actual FY19 Jan
```

recreate

Restores an environment to a clean state by re-creating the deployment.

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 Oracle 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.

  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

  Downgrading the deployment of Hybrid-enabled Essbase in EPM Enterprise Cloud Service environments is required if you are importing a snapshot from an environment that has Non-Hybrid Essbase.

  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 command 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 Migration or EPM Automate.
> - This command deletes migration history. As a result, the Migration Status Report available in Migration will not contain historic information.
> - Before using this command, perform a complete backup of the environment. You can create a backup snapshot by executing the runDailyMaintenance command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate recreate [-f] [removeAll=true|false] [EssbaseChange=upgrade|downgrade] [TempServiceType=Service_type]` where:

- `-f` forces the re-create process to start without user confirmation. If you do not use the `-f` option, EPM Automate prompts you to confirm your action.
- `removeAll`, optionally, removes all snapshots and the content of the inbox (uploaded files) and outbox (files exported from the environment). Default is `false`, which retains the snapshots and the content of inbox and outbox.
- `EssbaseChange`, optionally, upgrades or downgrades the current Essbase version. EPM Automate retains the current Essbase version if you do not specify this option. Permissible values are:
  - `upgrade` to change from Non-Hybrid Essbase to Hybrid Essbase
  - `downgrade` to change from Hybrid Essbase to Non-Hybrid Essbase.

  > **Caution:**
  >
  > Before using this option, read and understand the information available in About Essbase in EPM Cloud in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

- `TempServiceType`, optionally, converts 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 Free Form 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 *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*

  The behavior of this parameter is dependent on your subscription.

  - **Subscriptions other than EPM Standard Cloud Service and EPM Enterprise Cloud Service:**
    You can use the `TempServiceType` option to temporarily convert a Planning, Enterprise Planning, Tax Reporting, or Financial Consolidation and Close environment to an Account Reconciliation, Oracle Enterprise Data Management Cloud, or Profitability and Cost Management environment. For example, If you purchased a Planning environment, you can convert it to an Account Reconciliation environment by running the following command:

    ```
    epmautomate recreate -f removeAll=true TempServiceType=ARCS
    ```

    After converting the environment to Account Reconciliation, you can convert it to an Oracle Enterprise Data Management Cloud or Profitability and Cost Management environment by using the appropriate `TempServiceType` value. For example, to convert it to a Profitability and Cost Management environment, execute the following command:

    ```
    epmautomate recreate -f removeAll=true TempServiceType=PCMCS
    ```

To convert the environment back to the original service type, run the following command:

```
epmautomate recreate -f
```

Profitability and Cost Management: You can convert your Profitability and Cost Management environment to Planning, Enterprise Planning, or an Enterprise Profitability and Cost Management environment by running the following command:

```
epmautomate recreate -f removeAll=true TempServiceType=PBCS
```

To convert the environment back to the original Profitability and Cost Management environment, use the following command:

```
epmautomate recreate -f TempServiceType=PCMCS
```

> **Note:**
>
> Profitability and Cost Management environments cannot be converted to Account Reconciliation, Oracle Enterprise Data Management Cloud, or Narrative Reporting environments.

- **EPM Standard Cloud Service and EPM Enterprise Cloud Service subscriptions:** You can use the `TempServiceType` option to convert an EPM Cloud environment to any other supported environment.

  EPM Enterprise Cloud Service subscriptions use a common EPM Cloud platform. Initially, you can deploy any supported EPM Cloud business process.

  To switch from a deployed business process to another, you re-create the environment by specifying the new service type for the environment. For example, if you created an Account Reconciliation business process but now want to create an Oracle Enterprise Data Management Cloud environment, you run the re-create command as follows.

  ```
  epmautomate recreate -f removeAll=true TempServiceType=EDMCS
  ```

  To convert a business process (for example, Account Reconciliation) to Planning, Tax Reporting, or Financial Consolidation and Close, do not specify a `TempServiceType` value. For example, if you created an Account Reconciliation business process but now want to create a Planning Modules environment, you run the recreate command as follows.

  ```
  epmautomate recreate -f removeAll=true
  ```

Acceptable `TempServiceType` values:

- `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 environment

Examples

- Re-create the current environment, restore it to the original service type (if a recreate had been issued before with a `TempServiceType` parameter), and upgrade to Hybrid-enabled Essbase without removing user created snapshots and contents of inbox and outbox:

  ```
  epmautomate recreate -f EssbaseChange=upgrade
  ```

- Re-create the current environment and restore it to the original service type (if a recreate had been issued before with a `TempServiceType` parameter),, remove snapshots and the contents of inbox and outbox:
  ```
  epmautomate recreate -f removeAll=true
  ```

- Re-create the current environment as an Oracle Enterprise Data Management Cloud environment and remove the content of inbox and outbox, and existing snapshots:
  ```
  epmautomate recreate -f removeAll=true TempServiceType=EDMCS
  ```

- Re-create the current EPM Enterprise Cloud Service Account Reconciliation environment to a Financial Consolidation and Close environment and remove the content of inbox and outbox, and existing snapshots:
  ```
  epmautomate recreate -f removeAll=true
  ```

refreshCube

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

The time required to complete a cube refresh operation depends on the changes that you made to the application structure and the impact it has on the cube. For example, a refresh after updating a sparse block storage cube member may not take much time while a cube refresh after updating a dense block storage cube member or an aggregate storage cube member could take a considerable amount of time. You must ensure that the cube refresh operation is complete before the application is backed up during the next maintenance window.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate refreshCube [JOB_NAME]` where `JOB_NAME`, optionally, is the name of a Database Refresh job defined in the application.
Status of the operation is echoed in the console from which the command is run. You can also view the status from the **Recent Activity** page of the **Jobs** screen in the application.

Example

```
epmautomate refreshCube DaliyCubeRefresh
```

removeUserFromGroups

Removes the membership of a user from the Access Control groups identified in an ANSI or UTF-8 encoded CSV file.

The file format is as follows:

```
Group Name
Group1
Group2
```

> **Note:**
>
> These groups must exist in Access Control. Group Name values are not case-sensitive.

Use the uploadFile command to upload the file to an environment.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate removeUserFromGroups` *`FILE_NAME User_Login`* where:

- `FILE_NAME` is the name of a CSV file containing the names of the Access Control groups from which the user's membership is to be removed
- `User_Login` is the login ID of an Oracle Enterprise Performance Management Cloud user whose membership is to be removed from Access Control groups. This user login ID must exist in the identity domain that services the environment and must be assigned to a predefined role. This value is not case-sensitive.

Example

```
epmautomate removeUserFromGroups groups.CSV jdoe@examle.com
```

removeUsers

Deletes the accounts identified in an ANSI or UTF-8 encoded CSV file that was uploaded to the environment from an identity domain.

The file format is as follows:

```
User Login
jane.doe@example.com
jdoe@example.com
```

Use the uploadFile command to upload file to the environment. User Login values are not case-sensitive. For example, `jane.doe@example.com` is treated as being identical to `Jane.Doe@Example.com` or any variation in its case.

> **Note:**
>
> - 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.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

Usage

`epmautomate removeUsers FILE_NAME` where *`FILE_NAME`* is the name of a CSV file containing the login IDs of the users to be removed from the identity domain.

Example

```
epmautomate removeUsers remove_users.CSV
```

removeUsersFromGroup

Removes users listed in an ANSI or UTF-8 encoded CSV file from a group maintained in Access Control.

The file format is as follows:

```
User Login
jdoe
john.doe@example.com
```

User Login values are not case-sensitive. For example, `jane.doe@example.com` is treated as being identical to `Jane.Doe@Example.com` or any variation in its case. Use the uploadFile command to upload the file containing user logins to the environment.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate removeUsersFromGroup` *`FILE_NAME GROUP_NAME`* where:

- *`FILE_NAME`* is the name of a CSV file containing the login names of users you want to remove from a group maintained in Access Control.
- *`GROUP_NAME`* is the name of the Access Control group from which you want to remove users. This value is not case-sensitive.

> **Note:**
>
> User is removed from a group only if both 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 predefined role in the identity domain

Example

```
epmautomate removeUsersFromGroup user_file.CSV example_group
```

removeUsersFromTeam

Removes Oracle Enterprise Performance Management Cloud users listed in a CSV file from a team.

If a user included in the CSV file is not a member of the team, this command ignores that user. The values in this file are not case-sensitive. CSV file format of is as follows:

```
User Login
jdoe
jane.doe@example.com
```

Use the uploadFile to upload the .CSV file to the environment.

Applies to

Financial Consolidation and Close, Tax Reporting, and Account Reconciliation.

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate removeUsersFromTeam FILE.CSV TEAM_NAME` where:

- `FILE` identifies a UTF-8 formatted CSV file listing the login IDs of users to be removed from the team.
- `TEAM_NAME` identifies a team name as defined in Access Control. This value is not case-sensitive.

Example

```
epmautomate removeUsersFromTeam example_users.csv example_team
```

renameSnapshot

Renames a snapshot that you uploaded or a created in an environment.

If this command is executed to rename a snapshot that is in the process of being generated or archived, you will receive one of these errors:

- `File not found` if the snapshot is being generated
- `Archive process is in progress. Unable to Rename or Delete` if the snapshot is being archived

Do not rename the maintenance snapshot in an environment. To maintain a backup of the maintenance snapshot, you should download `Artifact Snapshot` from the environment to a local computer and then rename it as needed. See Overview of the Maintenance Snapshot in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate renameSnapshot SNAPSHOT_NAME NEW_SNAPSHOT_NAME` where:

- *SNAPSHOT_NAME* is the name of an existing snapshot. This value should not contain special characters such as space, \ (backslash), / (slash), * (asterisk), ? (question mark), " (quotation mark), < (less than), and > (greater than).
- *NEW_SNAPSHOT_NAME* is the unique name you want to assign to the snapshot.

Example

```
epmautomate renameSnapshot "Example Snapshot" Example_Snapshot_18_09_25
```

replay

Replays Oracle Smart View for Office, REST API, or EPM Automate load on an environment to enable performance testing under heavy load to verify that user experience is acceptable when the service is under specified load.

You must create a replay file that identifies the activities that should be executed on the service. See Preparing to Run the Replay Command for details information on how to create the replay file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate replay REPLAY_FILE_NAME.csv [duration=N] [trace=true] [lagTime=t] [encrypt=true|false]` where:

- *REPLAY_FILE_NAME* is a CSV file that stores the activities to be executed on the environment.
- `Duration`, optionally, indicates the number of minutes for which activities are executed on the environment.
  Activities in the HAR file are run one time if this value is not set. If the activities in the HAR file are completed within the time specified by this parameter, EPM Automate reruns the HAR file until the activities are complete. For example, assume that you set `duration=10` to replay a HAR file that takes three minutes to run. In this scenario, the replay command runs the HAR file activities four times (lasting 12 minutes) until the fourth iteration is complete.
- `trace=true` is an optional setting that instructs EPM Automate to create trace files in XML format.

If this optional setting is specified, EPM Automate creates one folder for each HAR file included in the replay CSV file and stores all related trace files in it. For each activity in the HAR file, EPM Automate generates one trace file that contains Smart View response. Trace files are named `trace-N.xml`; for example, `trace-1.xml` where `N` is a counter that starts at 1.

The folders that store the trace files are created in the directory from which EPM Automate is run. EPM Automate uses a combination of current system time of the environment and HAR file name in `YYYY_MM_DD_HH_MM_SS_HAR_FILE_NAME` format to name the folders. For example, if HAR file name is `forecast1.har`, the folder name may be `2016_06_08_10_21_42_forecast1`.

- `[lagTime=t]`, optionally, specifies the number of seconds that the command should wait between the execution of each HAR file included in the replay file. Default is 5 seconds. The command displays an error if you specify a value less than 5 seconds. Negative numbers (for example -1) and fractions (for example, 1/2) are not acceptable as the parameter value. Decimal values are supported.

  After initiating the execution of the first HAR file, the command waits for the number of seconds specified by this parameter to initiate the processing of the next HAR file. Because user activities are not usually initiated simultaneously, setting this parameter helps to create a more realistic simulation of load on an environment.

  For example, assume that you want to simulate the load of 1000 users signing on to an environment during the peak hour to perform activities. You can create HAR files to simulate these sessions and then run this command with a lag time of 6 seconds to replicate the load exerted on the environment.

- `encrypt=true|false`, optionally, specifies whether to encrypt all passwords included in the replay file. Default is `true`. A random encryption key is used to encrypt the password.

See A Sample Replay Session for detailed steps involved in executing this command.

Example

```
epmautomate replay forecast1.CSV duration=60 lagTime=5.6
```

resetService

Restarts the environment. You can, optionally, auto-tune the environment before restarting it to ensure that the Oracle Essbase index caches for Block Storage Option (BSO) cubes are optimized for your application.

By default, your environments are restarted right after the daily maintenance is completed. Auto-tuning your environment is important, for example, after importing a snapshot into an environment. Use this command only when you observe severe performance degradation or if you receive error messages indicating that the environment is unusable. Restarting an environment does not affect application customizations (for example, locale change, settings related to theme and currency, etc.). Restart takes up to 15 minutes.

Before using this command, ensure that business rules are not running in the environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate resetService "comment" [AutoTune=true|false] [-f]` where:

- `Comment` is a description of the issue that caused you to reset the environment. Comments must be enclosed in quotation marks.
- `AutoTune`, optionally, indicates whether to auto-tune the environment to optimize Essbase caches BSO cubes of your application. Default is `false`.

  Use this parameter only in environments that use Essbase BSO cubes: Planning (including Planning Modules), Financial Consolidation and Close, and Tax Reporting.
- `-f`, optionally, specifies that you want to force the restart of the environment without additional user interaction. If you do not use this option, EPM Automate prompts you to confirm your action. This option is useful if you schedule a script that uses this command.

Examples

- `epmautomate resetService "Users experience slow connections; force restarting the environment" -f`
- `epmautomate resetService "Users experience unacceptably slow connections"`
- `epmautomate resetService "Optimizing the Essbase cache" AutoTune=true`

restoreBackup

Copies an available backup snapshot of an OCI (Gen 2) environment so that it is available for import into the environment. This command does not work in Classic Oracle Enterprise Performance Management Cloud environments.

Use the listBackups command to determine if the backup you want to restore is available. Self-service restoration of a snapshot to the environment saves you processing time. After restoring the snapshot, use the importSnapshot command to import it into the environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmAutomate restoreBackup SNAPSHOT_NAME [targetName=TARGET_SNAPSHOT_NAME]`, where:

- *SNAPSHOT_NAME* is the name of a backup snapshot available in the environment as listed by the listBackups command.
- `targetName`, optionally, is the name for the backup snapshot, without an extension, in the target environment. If you do not specify this value, the backup snapshot is restored to the

target environment using the *SNAPSHOT_NAME*, but with a _ (underscore) replacing the / (slash). For example, if *SNAPSHOT_NAME* is `2022-05-14T00:08:56/Artifact_Snapshot.zip` the `targetName` will be `2022-05-14T00:08:56_Artifact_Snapshot.zip`.

Examples

- Services other than Narrative Reporting:
```
epmAutomate restoreBackup 2022-05-14T00:08:56/Artifact_Snapshot.zip
targetName=example_Artifact_Snapshot
```
- Narrative Reporting only:
```
epmAutomate restoreBackup 2022-02-16T21:00:02/EPRCS_Backup
targetName=Example_EPRCS_Backup
```

restructureCube

Performs a full restructure of a block storage cube to eliminate or reduce fragmentation. Restructuring also removes empty blocks, and will not push any changes from the application to the cube.

> **Note:**
>
> Before running this command, ensure that no one is using the application.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate restructureCube` *CUBE_NAME* where *CUBE_NAME* is the name of a cube exactly as it is in an application

Example

```
epmautomate restructureCube Plan1
```

roleAssignmentAuditReport

In OCI (Gen 2) environments, creates an audit report that lists the changes made to predefined and application role assignments over a period of time corresponding to the audit data retention period specified for your environment. The default retention period is 30 days. You can extend it to a maximum of 90 days by changing the **Audit Retention Period (days)** setting in the Oracle Cloud Identity Console. To retain the audit data for duration longer than 90 days, periodically download and archive this report and the Invalid Login Report.

The Role Assignment Audit Report lists the User Login Name for which a role change (in Action column) was made. It also includes the role that was assigned or unassigned, the user who performed the role change (Administrator column), and the timestamp (UTC) in 24-hour format when the action was completed.

	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 was 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

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management,Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator, or Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

Usage

`epmAutomate roleAssignmentAuditReport` *`FROM_DATE TO_DATE FILE_NAME`*`.CSV`, where:

- *`FROM_DATE`* indicates the start date (in `YYYY-MM-DD` format) of the period for which the report is to be generated. This date must fall within the audit retention period specified in the Oracle Cloud Identity Console.
- *`TO_DATE`* indicates the end date (in `YYYY-MM-DD` format) of the period for which the report is to be generated.
- *`FILE_NAME`* is the name of a CSV file for the report. You can download the generated report using the downloadFile command.

Example

```
epmAutomate roleAssignmentAuditReport 2021-06-01 2021-07-30 RoleAuditReport.CSV
```

roleAssignmentReport

Generates a Role Assignment Report (.CSV).

This report, by default, 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, optionally, can also be generated to list the Identity Domain Administrators of your environment. This report matches the CSV version of the Role Assignment Report generated from Access Control.

	A	B	C	D	E	F
1	User Login	First Name	Last Name	Email	Role	Granted through Group
2	Jdoe	John	Doe	jdoe@example.com	Planner	
3	jdoe	John	Doe	jdoe@example.com	Power User	
4	Jdoe	John	Doe	jdoe@example.com	Service Administrator	
5	jdoe	John	Doe	jdoe@example.com	Viewer	
6	Jdoe	John	Doe	jdoe@example.com	Mass Allocation	example->Power User
7	jdoe	John	Doe	jdoe@example.com	Run Integration	
8	jane.doe@example.com	Jane	Doe	jane.doe@example.com	Planner	
9	jane.doe@example.com	Jane	Doe	jane.doe@example.com	Power User	
10	jane.doe@example.com	Jane	Doe	jane.doe@example.com	Viewer	
11	jane.doe@example.com	Jane	Doe	jane.doe@example.com	Mass Allocation	example

Generating this report refreshes the user and role information available in Access Control.

For OCI (Gen 2) only: Oracle Enterprise Performance Management Cloud considers deactivated users as being identical to users not assigned to any predefined roles even though such users may have had predefined roles when they were deactivated. Information on deactivated users is not included in this report.

> **Note:**
>
> This command produces a report equivalent to that created using the provisionReport command.

You can download the report using the downloadFile command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate roleAssignmentReport` *`REPORT_NAME`*`.CSV [userType=IDAdmins|serviceUsers]`
where:

- *`REPORT_NAME`* is a name for the report.
- `userType`, optionally, identifies the type of users whose information is to be included in the report. Default is `serviceUsers`. Valid values are:
  - `serviceUsers` creates a report that contains information on all functional users (does not include Identity Domain Administrators if they are not assigned to a predefined role that grants access to the application).

  - `IDAdmins` creates a report that lists only the users assigned to the Identity Domain Administrator role.

Examples

- Generate the report listing only functional users:
  - `epmautomate roleAssignmentReport myReport.CSV`
  - `epmautomate roleAssignmentReport myReport.CSV userType=serviceUsers`
- Generate report listing only Identity Domain Administrators:
  `epmautomate roleAssignmentReport myReport.CSV userType=IDAdmins`

runAutomatch

Runs the Auto Match process to match transactions using the rules defined by a Service Administrator.

> **Note:**
>
> Run this command after you import transactions data into Transaction Matching using the importTmPremappedTransactions or the runDataRule command.

You can monitor the status of the auto match process on the **Job History** tab in Account Reconciliation.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate runAutomatch RECONCILIATION_TYPE` where `RECONCILIATION_TYPE` is a reconciliation type defined in Account Reconciliation.

Example

```
epmautomate runAutomatch INTERCOMPANY
```

runBatch

Executes a Data Management batch.

If batch execution mode in Data Management is set to Serial, control is returned when all the jobs in the batch are completed; if it is set to Parallel, control is returned when all jobs in the batch are submitted for execution.

> **Note:**
>
> This command cannot be used to execute direct data load integration from data sources into Oracle Enterprise Performance Management Cloud. Use the EPM Integration Agent to integrate direct data loads. For detailed information, see Performing a Direct Data Load using the EPM Integration Agent in *Administering Data Integration for Oracle Enterprise Performance Management Cloud*.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate runBatch BATCH_NAME` where *BATCH_NAME* is the name of a batch defined in Data Management.

Examples

```
epmautomate runBatch Accounting_batch
```

runBusinessRule

Launches a business rule.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

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

Usage

`epmautomate runBusinessRule RULE_NAME [PARAMETER=VALUE]` where:

- *RULE_NAME* is the name of a business rule exactly as it is defined in the environment.
- *PARAMETER=VALUE* indicates optional runtime parameters and their values required to execute the business rule.

> **Note:**
>
> - This command can execute a single business rule only. To execute a ruleset, use the runRuleSet command.
> - The rule is executed against the plan type to which it was deployed.
> - Default values are used if you do not provide values for runtime parameters. The command ignores runtime prompts that are not exact matches to those defined in the rule.
> - Use *PARAMETER=VALUE* pairing to specify as many runtime prompts as the business rule requires. The following example uses two runtime prompts (`Period` and `Entity`) and their values (`Q1` and `USA`).
>   See Specifying Multiple Values for a Parameter for information on entering multiple values for a parameter.

Example

```
epmautomate runBusinessRule RollupUSSales Period=Q1 Entity=USA
```

runCalc

Runs calculations in an application.

Using this command, you can run calculations using rules in a Model POV against data in a different Data POV without copying rules across POVs.

Applies to

Profitability and Cost Management

Required Roles

Service Administrator, Power User

Usage

`epmautomate runCalc` *`APPLICATION_NAME POV_NAME [DATA_POV_NAME]`* `PARAMETER=`*`VALUE`* `[comment="comment"] stringDelimiter=`*`"DELIMITER"`* where:

- *`APPLICATION_NAME`* is the name of the Profitability and Cost Management application that contains the POV to be calculated.
- *`POV_NAME`* is the name of the model POV to be calculated.
- *`DATA_POV_NAME`* is, optionally, the name of the data POV that is to be calculated using the rules of the model POV.

  If *`DATA_POV_NAME`* is not specified, by default, *`POV_NAME`* will be used.

  You can use only `exeType=ALL_RULES` if you specify *`DATA_POV_NAME`*.
- `PARAMETER=`*`VALUE`* indicates runtime parameters and their values to run the calculation. Specify as many parameter and value pairings as the process requires. Valid parameters and their values:
  - `exeType=ALL_RULES|RULESET_SUBSET|SINGLE_RULE` identifies the rule execution type. This is a required parameter.

Depending on the value set for `exeType`, the following parameters may be specified:

  * If you specify `exeType=ALL_RULES`, do not include rule subset or single rule related parameters such as `subsetStart`, `subsetEnd`, `ruleSetName`, and `ruleName`. Must use this `exeType` if you set *`DATA_POV_NAME`* parameter.
  * If you specify `exeType=SINGLE_RULE`, specify the values for `ruleSetName` and `ruleName` only.
  * If you specify `exeType=RULESET_SUBSET`, specify the values for `subsetStart` and `subsetEnd`.

- `subsetStart` specifies the sequence number of the first rule in the rule set to run
- `subsetEnd` specifies the sequence number of the last rule in the rule set to run
- `ruleSetName` identifies the rule set that contains the calculations you want to run
- `ruleName` name of the rule to run (to run a single rule)
- `isClearCalculated=true|false` specifies whether to clear existing calculations
- `isExecuteCalculations=true|false` specifies whether to run calculations
- `isRunNow=true|false` set this value to `true` to run the process now
- `optimizeReporting=true|false` set this optional value to `false` if the calculations are to be run without optimization for reporting. Default is `true`
  Best Practice:
  * Set `optimizeReporting=false` only when necessary to save processing time; for example, when running a single rule or a sequential series of several POVs
  * When running multiple concurrent calculation jobs, set `optimizeReporting=true` for all jobs; only the last job to complete will perform the aggregation, avoiding redundant processing and preventing running jobs from slowing down.

> **Note:**
>
> Parameter values (`true` or `false`) must be in all lower case.

- `comment` is an optional comment enclosed in double quotation marks
- `stringDelimiter` is the delimiter used in POV values. Delimiter must be enclosed in double quotation marks.

Example

```
epmautomate runCalc BksML12 2012_Jan_Actual Jan-2016 isClearCalculated=true
isExecuteCalculations=true isRunNow=true subsetStart=10 subsetEnd=20
ruleSetName="Utilities Expense Adjustment" ruleName="Occupancy Expense
Allocations" exeType="ALL_RULES" comment="Test calculation" stringDelimiter="_"
```

runComplianceReport

Generates a report that is defined in Reconciliation Compliance.

See these information sources in *Administering Account Reconciliation*:

- Using Reports for instructions on defining reports.

- Generating Predefined Reports in Reconciliation Compliance for a list of predefined Reconciliation Compliance reports and the parameters for generating them.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate runComplianceReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]` where:

- `FILE_NAME` is a unique file name for the report that will be generated. If a report with this name exists on the server, it will be overwritten. Use thedownloadFile command to download this report to a local computer.
- `GROUP_NAME` is the name of the group with which the report is associated.
- `REPORT_NAME` is a unique name for the report to be generated.
- `REPORT_FORMAT` is one of the following formats for the report:
  - PDF
  - HTML (not supported for graphs and charts)
  - XLSX (not supported for graphs)
  - CSV
  - CSV2

> **Note:**
>
> *`REPORT_FORMAT`* `CSV` does not permit the formatting of data based on a template while `CSV2` does. Generating `CSV2` formatted report takes more time compared to `CSV` output.

- `Param=value`, optionally identifies the required parameters for generating the report. For example, the `Balance By Account Type` report takes two parameters `Period` with the value `July 2017` and `Currency Bucket` with the value `Entered`. You should specify these parameters as `"Period=July 2017" "Currency Bucket=Entered"`.

Example

```
epmautomate runComplianceReport "Example_File Name""Reconciliation Manager"
"Balance By Account Type" PDF "Period=July 2017" "Currency Bucket=Entered"
```

runDailyMaintenance

Starts the daily service maintenance process right away instead of waiting for the scheduled daily maintenance window.

This command enables you to force the creation of a backup snapshot and to update your environment. Before running this command, ensure that no one is using the environment. Daily maintenance schedule is not affected by this command. You use this command if you do not want to wait for the next maintenance window for changes to the environment to take effect, for example, after applying a one-off patch.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate runDailyMaintenance [skipNext=true|false] [-f]` where:

- `skipNext`, optionally, indicates whether to skip the next occurrence of the daily maintenance process. Default is false.
- `-f`, optionally, indicates whether to force start the maintenance process without user confirmation. If you do not use the `-f` option, EPM Automate prompts you to confirm your action.

Examples

- To force start an off cycle daily maintenance without skipping next scheduled maintenance:
  ```
  epmautomate runDailyMaintenance -f
  ```
- To force start an off cycle daily maintenance and skip the next scheduled maintenance:
  ```
  epmautomate runDailyMaintenance skipNext=true -f
  ```
- To start an off cycle daily maintenance and skip the next scheduled maintenance:
  ```
  epmautomate runDailyMaintenance skipNext=true
  ```

runDataRule

Executes a Data Management data load rule based on the start period and end period, and import or export options that you specify.

> **Note:**
>
> This command cannot be used to execute direct data load integration from data sources into Oracle Enterprise Performance Management Cloud. Use the EPM Integration Agent to integrate direct data loads. For detailed information, see Performing a Direct Data Load using the EPM Integration Agent in *Administering Data Integration for Oracle Enterprise Performance Management Cloud*.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate runDataRule` *`RULE_NAME START_PERIOD END_PERIOD IMPORT_MODE EXPORT_MODE [FILE_NAME]`* where:

- *`RULE_NAME`* is a name of a data load rule defined in Data Management. You should enclose the rule name in quotation marks if it contains space.
- *`START_PERIOD`* is the first period for which data is to be loaded. This period name must be defined in Data Management period mapping.
- *`END_PERIOD`* is, for multi-period data load, the last period for which data is to be loaded. For single period load, enter the same period as start period. This period name must be defined in Data Management period mapping.
- *`IMPORT_MODE`* determines how the data is imported into Data Management.

  Import mode settings are case-sensitive. Acceptable values are:

  - `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 recalculate the data
  - `NONE` to skip data import into Data Management staging table
- *`EXPORT_MODE`* determines how the data is exported to the application.

  Export mode settings are case-sensitive. Acceptable values are:

  - `STORE_DATA` to merge the data in the Data Management staging table with the existing data. Always use this export option in the Data Management jobs used to load metadata.
  - `ADD_DATA` to add the data in the Data Management staging table to the application.
  - `SUBTRACT_DATA` to subtract the data in the Data Management staging table from existing 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 the application.

> **Note:**
>
> For Financial Consolidation and Close, only these export modes are supported:
>
> - `MERGE` to merge the data in the Data Management staging table with the existing data
> - `REPLACE` to remove entries from DM staging table and replace with those from the data load
> - `NONE` to skip data export from Data Management to the application
>
> For Oracle Fusion Cloud as a target, only these export modes are supported:
>
> - `MERGE` to merge the data in the Data Management staging table with the existing data
> - `NONE` to skip data export from Data Management to the application

- *`FILE_NAME`* is an optional file name. If you do not specify a file name, EPM Automate imports the data contained in the file name specified in the load data rule. This file must be available in the inbox folder or in a folder within it.

  When loading Bank Administration Institute (BAI) format files for Account Reconciliation, do not specify a value for this parameter. You must always specify the file name for loading BAI files in the data rule definition.

> **Note:**
>
> If a path is specified in the data rule, do not specify the file path in the command; specify only the file name. If a path is not specified in the data rule; specify the full path to the data file.

Examples

- Multi-period Import:
  ```
  epmautomate runDataRule VisionActual Mar-15 Jun-15 REPLACE STORE_DATA inbox/
  Vision/GLActual.dat
  ```
- Single-period Import:

  ```
  epmautomate runDataRule "Vision Actual" Mar-15 Mar-15 REPLACE STORE_DATA
  inbox/Vision/GLActual.dat
  ```

runDMReport

Creates a Data Management report and stores it in the `outbox/reports` folder.

The generated report is named based on the ID of the Data Management job that generates the report and the report format. For example, if the report job ID is 2112 and the report output format that you specify is PDF, the report name is `2112.pdf`. The report name is displayed in the console after the report is generated. You can also identify the report name from the Process Details tab in Data Management or by using the listFiles command.

Use the downloadFile command to download the report to a local computer.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

`epmautomate runDMReport` *`REPORT_NAME PARAMETER=Value`* `"Report Output Format=[PDF|HTML|XLS|XLSX]"` where:

- *`REPORT_NAME`* is the name of the Data Management report template to be used for generating the report.
- *`PARAMETER=Value`* indicates report parameters and their values. You specify as many parameters as required in `PARAMETER=Value` format. The list of required parameters depends on the report that you want to generate.

> **Note:**
>
> Report run time parameters are defined when you design your reports. To run this command, you must generate and copy these parameters and values to EPM Automate from the Workflow tab. To generate runtime parameters of a report, in the Workflow tab of Data Management, click **Report Execution** and then select a group from **Report Group**. Select the report for which you want to generate the parameters, then click **Create Report Script**. Optionally, specify report parameter values, then select an output format, and then click **OK**. Use the parameters shown in **Generate Report Script** to specify runtime parameters and values to generate the report

- `Report Output Format` indicates the report output format. Valid options are PDF, HTML, XLS, and XLSX. The default report format is PDF.

Example

```
epmautomate runDMReport "TB Current Location By Target Acct (Cat,Per)"
"Period=Jul 14" "Category=Forecast" "Location=FCSTtoVISCONSOL1" "Rule
Name=FCSTtoVISCONSOL1" "Report Output Format=HTML"
```

runIntegration

Executes a Data Integration job to import data into an Oracle Enterprise Performance Management Cloud business process or export data from a business process to an external system.

This command deprecates the runDataRule command. Oracle recommends that you start using this command instead of therunDataRule command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User

Usage

```
epmautomate runIntegration JOB_NAME importMode=Append|Replace|"Map and
Validate"|"No Import"|Direct exportMode=Merge|Replace|Accumulate|Subtract|"No
Export"|Check periodName={PERIOD_NAME} [inputFileName=FILE_NAME] [PARAMETERS]
```

- For Standard Mode integrations, you must specify the values for `importMode`, `exportMode`, and `periodName`
- For Quick Mode integrations, you must specify the value for `exportMode`
- Parameter names and their values are case-sensitive

In this command:

- *JOB_NAME* is the name of an integration job defined in Data Integration.
- `importMode` determines how the data is imported into Data Integration. Acceptable import modes are:
  - `Append` to add to the existing POV data in Data Integration.
  - `Replace` to delete the POV data and replace it with the data from the file.
  - `Map and Validate` to skip data import, but reprocess the data with updated mappings and logic accounts.
  - `No Import` to skip data import into the Data Integration staging table.
- `exportMode` determines how the data is loaded to the target application. For Quick Mode integrations, you cannot use `Check` and `No Export` as the value of `exportMode` parameter. Acceptable export mode values are:
  - `Merge` to update the existing data and add new data.
  - `Replace` to clear the existing data in the POV and load it with new data. For Standard mode, the data is cleared for Scenario, Version, Year, Period, and Entity dimensions. For Quick Mode, data is cleared for Year, Period, and Entity dimensions. You can define custom clear regions for both modes.
  - `Accumulate` to add the data to the existing data. Applicable to Planning, Planning Modules, Financial Consolidation and Close, Tax Reporting, Profitability and Cost Management, and Enterprise Profitability and Cost Management.
  - `Subtract` to subtract the data from existing balance. Applicable to Profitability and Cost Management, and Enterprise Profitability and Cost Management.
    For Quick Mode integrations:
    * You cannot use `Check` and `No Export` as the value of this parameter.
    * For Planning, Planning Modules, and Financial Consolidation and Close, only valid values are `Replace`, `Merge`, and `Accumulate`.

  - `No Export` to skip data export. Use this mode to load data into the staging table for review before loading to the target application.
  - `Check` to only perform a data validation check.

    For Oracle Fusion Cloud as a target, only these export modes are supported:

    * `MERGE` to merge the data in the Data Integration staging table with the existing data
    * `NONE` to skip data export from Data Integration to the application

- `periodName` is the name of one ore more periods or period ranges, each enclosed in curly brackets, for which to import or export the data. Acceptable period naming conventions are as follows:
  - For single period loads, specify the period name enclosed in curly brackets, for example, `{Jan-21}`
  - For multi-period loads, enclose start and end period names in curly brackets, for example, `{Jan-21}{Mar-21}` (to load data for all periods starting Jan-21 and ending Mar-21)
  - **For Planning, Planning Modules, Financial Consolidation and CloseFreeForm, and Tax Reporting:** You can specify the Business Process Period Name and year in the format `{Jan#FY21}{Mar#FY21}` to load data for all periods starting Jan-21 and ending Mar-21.
    Period name must be enclosed in curly brackets.

    * Single Period—Refers to the Data Management period name for a single period defined in Period mapping.
    * Multi-Period—Refers to a multi-period load. The parameter is specified in `{Month-Year}{Month-Year}` format. For example, {Jan-20}{Mar-20} for a multi-period load from Jan-20 to Mar-20.
    * Planning Period Name—Refers to a Planning period name in `{Month#Year}` format, for example, {Jan#FY20}{Mar#FY20}. Using this convention, you do not need to specify Data Integration period names. Instead you specify the member names for the Year and Scenario dimensions.
      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.

      Using this convention is useful if triggered from an EPM 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.
    * Substitution Variable—This is an extension of the preceding Planning period name format whereby a substitution variable, instead of the actual Year and Month member names, may be specified in `{Month#&CurYr}{&FcstMonth#&CurYr}` format; for example, `{Jan#&CurYr}{&FcstMonth#&CurYr}`.
      A combination of both actual member names as well as substitution variables is supported.

      This format is supported in the Planning,Tax Reporting, and Financial Consolidation and Close business processes.

      The application period mapping or global period mapping must exist in the Data Integration of the environment where the command is run, with the Year and Month values available 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.

* GLOBAL POV—Executes the data load for the Global POV period. Use the format `{GLOBAL_POV}`.

> **Note:**
>
> If you use any period naming parameter other than the parameters described in this discussion, you'll get an `Invalid Input – HTTP 400` error message.

– `{GLOBAL_POV}` to execute the data load for the period defined in Global POV in the system or on the Application Settings in Data Integration.

> **Note:**
>
> {Month#Year} period naming convention format is supported for Planning, Planning Modules, Financial Consolidation and Close, and Tax Reporting. Under this convention, you can specify member names for the Year and Scenario dimensions instead of the Data Integration period names. This approach is useful if the command is triggered from a Groovy script by capturing the Year and Period member names.
> The `{Jan#&CurYr}{&FcstMonth#&CurYr}` substitution variable naming convention is an extension of the preceding period naming convention. You can specify substitution variable instead of the Year and Month member names if you are running this command against Planning, Planning Modules, Financial Consolidation and Close, and Tax Reporting. A combination of member names and substitution variables is also supported.
>
> The preceding period naming and substitution variable naming conventions work only if application period mappings or global period mappings with the Year and Month in the target values already exist in Data Integration.

- `inputFileName`, for file-based data loads, specifies the name of the file, available in the inbox, from which data is to be imported. If you specify the directory name in the Integration definition, then pass only the file name. If you do not include a directory name in the Integration definition, use `inbox/`*`DIR_NAME/FILE_NAME`* format, for example, `inbox/GLBALANCES.txt` or `inbox/EBSGL/GLBALANCES.txt`. If the file has been uploaded to the default location in the environment, use `#epminbox/`*`FILE_NAME`* convention, for example, `#epminbox/GLBALANCES.txt`, to identify the input data file.
  This parameter is applicable only to native file-based data loads. If you do not specify this parameter value for file-based data loads, this command imports data from the file specified in the integration definition. If you specify this parameter value for data loads that are not file-based, the command ignores it.
- *`PARAMETERS`*, optionally, identifies runtime parameters in *`PARAMETER_NAME="PARAMETER"`* format. Parameters include both source filters and target options.

> **Note:**
>
> The only parameter that you can use at this time for a dimension (metadata) type of target application is `"Refresh Database"=Yes|No`.

Examples

- Single period import:
  ```
  epmAutomate runIntegration VisionDataLoad importMode=Replace exportMode=Merge
  periodName="{Mar-15}"
  ```
- Multi-period Import:
  ```
  epmAutomate runIntegration VisionDataLoad importMode=Replace exportMode=Merge
  periodName="{Mar-15}{Jun-15}"
  ```
- Incremental file-based data integration:
  ```
  epmAutomate runIntegration IncrementalFileLoad importMode=Replace
  exportMode=Merge periodName="{Jan-20}{Mar-20}" inputFileName=File1.txt
  ```

runIntercompanyMatchingReport

Generates an Intercompany Matching report that helps you correctly match the transactions between related entities and partners as a part of the consolidation process and for analysis and auditing purposes.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator, Power User, User
Users with Power User or User must be granted additional security through ACL

Usage

`epmautomate runIntercompanyMatchingReport` *`JOB_NAME FILE_NAME`* `[scenario=scenario] [years=years] [period=period_name] [ReportFormat=HTML]` where:

- `JOB_NAME` is the name of an Intercompany report job definition existing in the application. See Managing Intercompany Matching Reports in *Working with Financial Consolidation and Close* for more information. This command generates the Intercompany Matching report using the settings available in the job. Be sure to select Outbox as the location for the generated report.

  > **Note:**
  >
  > You can override the scenario, years, period, and report format settings specified in the job by entering these optional values while running this command.

- `FILE_NAME` is the name of the report file. This report is generated in the outbox. Use the downloadFile command to download it to a local computer or the sendMail command to email it.
- `SCENARIO`, optionally, is a name of a scenario, defined in the application, for which the report is to be generated.
- `YEARS`, optionally, is the year for which the report is to be generated.
- `PERIOD`, optionally, is the period for which the report is to be generated.
- `ReportFormat`, optionally, is the format for the report. Acceptable values are HTML, PDF, and XLSX.

> **Note:**
>
> - You can generate this report for a combination of one scenario, year, and period. Do not specify multiple scenarios, years, or periods.
> - If you do not specify values for scenario, years, period, and report format, the corresponding values specified in the report job definition (identified by `JOB_NAME`) are used. If specified, these optional values will override the values set in the report job definition.

Examples

- Creating the report by overriding the values set in the job definition:
  ```
  epmautomate runIntercompanyMatchingReport IC_Job_01 SampleICReport.html
  scenario=Actual years=FY22 period=Jan reportFormat=HTML
  ```
- Creating the report using the values set in the job definition:
  ```
  epmautomate runIntercompanyMatchingReport IC_Job_01 SampleICReport
  ```

runMatchingReport

Generates a report that is defined in Transaction Matching.

See Generating Predefined Reports in Transaction Matching in *Administering Account Reconciliation* for a list of predefined Transaction Matching reports and the parameters for generating them.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate runMatchingReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]` where:

- `FILE_NAME` is a unique file name for the report that will be generated. If a report with this name exists on the server, it will be overwritten. Use the downloadFile command to download this report to a local computer.
- `GROUP_NAME` is the name of the group with which the report is associated.
- `REPORT_NAME` is a unique name for the report to be generated.
- `REPORT_FORMAT` is one of the following formats for the report:
  - PDF
  - HTML (not supported for graphs and charts)
  - XLSX (not supported for graphs)
  - CSV

  - CSV2

> **Note:**
>
> *REPORT_FORMAT* `CSV` does not permit the formatting of data based on a template while `CSV2` does. Generating `CSV2` formatted report takes more time compared to `CSV` output.

- `Param=Value`, optionally identifies the required parameters for generating the report. For example, for the `Match Type Configuration` report which takes the parameter `status` with the value `approved`, specify the parameter and value as `status=Approved`.

Example

```
epmautomate runMatchingReport Example_FileName "Transaction Matching" "Match Type Configuration" HTML "status=Approved"
```

runPlanTypeMap

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 job of type `plan type map`.

Applies to

Planning, Planning Modules, FreeForm, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate runPlanTypeMap` *`JOB_NAME`* `[clearData=true|false]` where:

- `JOB_NAME` is the name of a job of type `plan type map` defined in the application.
- `clearData` is an optional setting that indicates whether the data in the target database should be removed before copying data. If this parameter value is not set, the default value `true` is used.

  Parameter values (`true` or `false`) must be in all lower case.

Example

```
epmautomate runPlanTypeMap CampaignToReporting clearData=false
```

runRuleSet

Launches a business ruleset.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

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

Usage

`epmautomate runRuleSet RULESET_NAME [PARAMETER=VALUE]` where:

- *`RULESET_NAME`* is the name of a business ruleset exactly as defined in the environment.
- *`PARAMETER=VALUE`* indicates optional runtime parameters and their values required to execute the ruleset.

> **Note:**
>
> The ruleset is executed against the plan type to which it is deployed.
>
> Use *`PARAMETER=VALUE`* pairing to specify as many runtime prompts as the ruleset requires. The following example uses two runtime prompts (`Period` and `Entity`) and their values (`Q1` and `USA`).
>
> Default values are used if you do not provide values for runtime parameters. The command ignores runtime prompts that are not exact matches to those defined for the ruleset.
>
> See Specifying Multiple Values for a Parameter for information on entering multiple values for a parameter.

Example

```
epmautomate runRuleSet RollupUSSales Period=Q1 Entity=USA
```

runSupplementalDataReport

Generates relational reports that display data from Supplemental Data Manager.

Supplemental Data Reports are grouped as Non-Consolidation Reports in Financial Consolidation and Close and Tax Reporting. See "List of Predefined Reports and Parameters" section in Generate Report for Financial Consolidation and Close and Tax Reporting in *REST API for Oracle Enterprise Performance Management Cloud* for a list of reports you can generate and the parameters for generating them.

Applies to

Financial Consolidation and Close, and Tax Reporting.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate runSupplementalDataReport FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT [Param=value]` where:

- `FILE_NAME` is a unique file name for the report.
- `GROUP_NAME` is the name of the group with which the report is associated.

- `REPORT_NAME` is a unique name for the report to be generated.
- `REPORT_FORMAT` is one of the following formats for the report:
  - PDF
  - HTML (not supported for graphs and charts)
  - XLSX (not supported for graphs)
  - CSV
  - CSV2

  `REPORT_FORMAT` `CSV` does not permit the formatting of data based on a template while `CSV2` does. Generating `CSV2` formatted report takes more time compared to `CSV` output.
- `Param=value`, optionally identifies the required parameters for generating the report. For example, to generate the `At Risk Tasks` report, which takes a `schedule name` with the value `monthly` and a `period` with the value `Jan`, specify `"schedule name"=monthly period=Jan`.

Example

```
epmautomate runSupplementalDataReport Example_File_name Group1 "At Risk Tasks"
html "schedule name"=monthly period=Jan
```

runTaskManagerReport

Generates relational reports that display data from Task Manager.

Task Manager reports are grouped as Non-Consolidation Reports in Financial Consolidation and Close and Tax Reporting.

See "List of Predefined Reports and Parameters" section in Generate Report for Financial Consolidation and Close and Tax Reporting in *REST API for Oracle Enterprise Performance Management Cloud* for a list of reports you can generate and the parameters for generating them.

Applies to

Financial Consolidation and Close, and Tax Reporting.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

`epmautomate runTaskManagerReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]` where:

- `FILE_NAME` is a unique file name for the report.
- `GROUP_NAME` is the name of the group with which the report is associated.
- `REPORT_NAME` is a unique name for the report to be generated.
- `REPORT_FORMAT` is one of the following formats for the report:
  - PDF
  - HTML (not supported for graphs and charts)

  - XLSX (not supported for graphs)
  - CSV
  - CSV2

> **Note:**
>
> *REPORT_FORMAT* `CSV` does not permit the formatting of data based on a template while `CSV2` does. Generating `CSV2` formatted report takes more time compared to `CSV` output.

- `Param=value`, optionally identifies the required parameters for generating the report. For example, to generate the `Early Tasks` report, which takes a `schedule name` with the value `monthly` and a `period` with the value `Jan`, specify `"schedule name"=monthly period=Jan`.

Example

```
epmautomate runTaskManagerReport Example_File_name Group1 "Early Tasks" PDF
"schedule name"=monthly period=Jan
```

sendMail

Sends an email, with the option to attach files from Oracle Enterprise Performance Management Cloud.

You can incorporate this command into scrips to notify users of various conditions or to send reports.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management,Enterprise Profitability and Cost Management, Narrative Reporting, Oracle Enterprise Data Management Cloud, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate sendMail ToAddress Subject [Body="MessageBody"] [Attachments=`*`FILE1,FILE2`*`]` where:

- `ToAddress` identifies semicolon separated email addresses of recipients enclosed in double quotes. Example, `"jdoe@example.com;jane.doe@example.com"`.
- `Subject` identifies the email subject.
- `Body="MessageBody"`, optionally, is the email content. If not specified, there is no body to the email.

> **Note:**
>
> Use valid HTML tags to format the message body to create a desired email format. The entire message body (including all HTML tags) must be specified as one line and should not contain new line charracters. See Examples.

- `Attachments`, optionally, identifies a comma separated list of files available in EPM Cloud to be attached to the email. For example, `outbox/errorFile.txt,inbox/users.csv`.

> **Note:**
>
> - Use * (asterisk) as the wildcard for one character in the file name. For example, specify `outbox/user*.csv` to attach all files in the outbox with five letter file names that fit the pattern.
> - You can attach any file, other than snapshots, listed by the listFiles command, as an email attachment. The size of the attachment must not exceed 10 MB.

Examples

- **Unformatted Email:** `epmautomate sendMail "jdoe@example.com;jane.doe@example.com" "Data Load Process Failed" Body="Data Load 1 Failed" Attachments=outbox/Errorfile.txt,outbox/Errofile2.txt`
- **Formatted Email:** `epmautomate sendMail jdoe@example.com "Send Formatted Email" "Body=<!DOCTYPE html><html><body><h1>EpmAutomate Email Formatting</h1><p>Hi,</p><p>Test Allocation Rules, Volume, and SPT data were loaded into FY22_Feb_Actual_Version POV.</p><p>Check the attachment for details.</p></body></html>" Attachments=outbox/loadResults.txt`

setApplicationAdminMode

Places the application in administration mode so that access to the application is limited to Service Administrators only.

This command is useful to prevent users from working on the application when Service Administrators are performing administrative operations. The application remains in administration mode until you change it back so that all users can access it.

> **Note:**
>
> This command replaces the applicationAdminMode command, which has been deprecated, but not yet removed from EPM Automate.

Use the getApplicationAdminMode command to check the current status of the environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Account Reconciliation, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

```
epmautomate setApplicationAdminMode true|false
```

In this command, specify `true` to place the application in administration mode and `false` to return it to normal mode so that all users can access it

Examples

- Place the application in administration mode:
  `epmautomate setApplicationAdminMode true`
- Return the application to normal operation:
  `epmautomate setApplicationAdminMode false`

setDailyMaintenanceStartTime

Sets the time (UTC or another time zone) at which the daily maintenance of the environment starts. The maintenance of the environment need not start at the top of the hour; you can set the hour and the minute at which the maintenance should start.

To ensure that the use of this command does not interfere with Oracle's requirement for creating backups, this command will not change the maintenance start time if the daily maintenance process did not run in the last 36 hours.

> **Note:**
>
> Service Administrators who are currently logged in to the environment using a browser will see the new daily maintenance start time only after they sign out and then sign in.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setDailyMaintenanceStartTime` *`StartTime`* where *`StartTime`* is the time (in HH:MM 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 - 23:59. If the start time is not to be set in UTC, specify a valid standard time zone; for example, `"14:35 America/Los_Angeles"` for 2:35 pm Pacific Standard Time.

Examples

- Set daily maintenance start to 2:20 PM UTC:
  ```
  epmautomate setDailyMaintenanceStartTime 14:20
  ```
- Set daily maintenance start to 2:35 PM Pacific Standard Time:
  ```
  epmautomate setDailyMaintenanceStartTime "14:35 America/Los Angeles"
  ```

setDemoDates

Updates Oracle internal demo data as needed.

Use this command only on installations setup with Oracle internal demo data.

Account Reconciliation only: This command resets dates for all reconciliations that have associated `Demo Code` attribute values of `setdemodates` or `setdemodatesnostatuschange`. This command handles reconciliations for up to 12 periods: a current period and 11 prior (historic) periods. If reconciliations from more than two periods are tagged with the `Demo Code` attribute, the command treats those periods as being in the prior period. Reconciliations that do not have this attribute value are not affected.

- If the value is `setdemodates`, the command resets the reconciliation dates based on the specified date and a random status
- If the value is `setdemodatesnostatuschange`, the command resets the reconciliation dates based on the specified date without changing the reconciliation status

Financial Consolidation and Close and Tax Reporting only: This command resets the tasks start and end dates, and other related date information, to make the tasks look good for a demo. It calculates the new task dates based on the value of the `SETDEMODATES` attribute set in the task's schedule along with the value of the `Demo Date` value that you provide. If `Demo Date` value is not specified, the command uses today's date to calculate the new task dates.

> **Note:**
>
> Tasks in schedules that do not have `SETDEMODATES` value are not affected.

Based on the `Demo Date` that you specify, this command moves forward all dates associated with the task. This includes core run time dates (start date, end date, etc.) and ancillary dates including those of history, individual workflow due dates, and start date (actual). Task status is not affected.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, and Account Reconciliation, Sales Planning, and Strategic Workforce Planning

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate setDemoDates [DEMO_DATE]` where *`DEMO_DATE`* is an optional date in `YYYY-MM-DD` format. Reconciliations are reset to the current date if you do not specify this value.

Example

```
epmautomate setDemoDates 2020-02-15
```

setEJJournalStatus

In Financial Consolidation and Close, sets the Enterprise Journal posting result from the ERP System. Use this command to update the posting status of Journals that are in `Post in Progress` status irrespective of their workflow status.

This command uses a CSV file identifying the status of import into the ERP system. You use the uploadFile command to upload the import file to the environment. The CSV file format is as follows:

```
Year,Period,Journal ID,Posting Status,Message
2020,Dec,1000001021,Posted,"SUCCESS"
2020,Dec,1000001022,Failed,"Row Header No: 2,10000415 - Linked value 6 does
not exist Application-defined or object-defined error 65171"
2020,Dec,1000001022,Failed,"Row Header No: 7,10000415 - Z_ECS_MSG (001)Enter
a valid account number"
2020,Dec,1000001022,Failed,"Row Header No: 7,10000415 - Z_ECS_MSG (002) Enter
a valid cost center"
```

Message column is optional and may be omitted.

This command does not export Enterprise Journal data from Financial Consolidation and Close or import it into the ERP system.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator

Usage

`epmautomate setEJJournalStatus` *`FILE_NAME`*`.csv` where *`FILE_NAME`* identifies the CSV file containing the status of import into the ERP system.

Example

```
epmautomate setEJJournalStatus JournalStatus.csv
```

setEncryptionKey

Sets a custom encryption key for database access.

Using this command provides a Bring Your Own Key (BYOK) solution for customers to include Oracle Enterprise Performance Management Cloud in their standard key management rotation.

The custom encryption key takes effect after the next daily maintenance of the environment. You can activate it immediately by running the resetService command.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setEncryptionKey key=`*`key`* where *`key`* is a custom string of any length that you want to use as the encryption key.

Examples

- Set encryption key: `epmautomate setEncryptionKey key=se!m+a2J`
- Remove encryption key: `epmautomate setEncryptionKey key=`

setEssbaseQryGovExecTime

Sets the maximum amount of time, in seconds, that an Oracle Essbase query can use to retrieve and deliver information before the query is terminated.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Profitability and Cost Management, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmautomate setEssbaseQryGovExecTime` *`TIME`* where `TIME` identifies the number of seconds after which Essbase queries are to be terminated. This value must be a whole number not exceeding 70000.

Oracle recommends that you do not set this value to 0 (zero) to prevent Essbase queries from running indefinitely.

Example

```
epmautomate setEssbaseQryGovExecTime 600
```

setIdleSessionTimeout

Changes the session timeout (in minutes) of the Oracle Enterprise Performance Management Cloud environment. The new session timeout becomes active after the next daily maintenance of the environment. Use this command to change the default session timeout (75 minutes) to a different value. After a session is idle for the duration specified using this command, the user is redirected to the Login page.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setIdleSessionTimeout` *`MINUTES`*, where MINUTES identifies the new idle session timeout (minimum 15 and maximum 150) in minutes.

Example

```
epmautomate setIdleSessionTimeout 30
```

setIPAllowlist

For OCI (Gen 2) environments, configures an allowlist of IP addresses and Classless Inter-Domain Routings (CIDRs) that are permitted to access Oracle Enterprise Performance Management Cloud. This command adds or removes IPv4 addresses and CIDRs.

This command provides a self-service method to configure an allowlist for EPM Cloud environments hosted on OCI (Gen2).

> **Note:**
>
> - This command cannot be used to configure allowlist in Classic environments. For Classic environments, use the Service Details screen of My Services (Classic) to create allowlist or denylist rules to regulate how users access EPM Cloud environments.
> - When you setup IP allowlist for an EPM Cloud environment, you permit connections only from those specific IP addresses. In this scenario, requesting access from another EPM Cloud will not work unless you add the outbound IP addresses of the data center or region where the requesting environment is located to the IP allowlist. See Outbound IP Addresses of EPM Cloud Data Centers and Regions in *Oracle Enterprise Performance Management Cloud Operations Guide* for the IP addresses that you need to add to ensure that the other environments can communicate with the environment for which the IP allowlist is being set up.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmAutomate setIPAllowlist add|remove` *`FILE_NAME`*`.txt` where:

- `add` adds the IP addresses and CIDRs listed in a text file to the allowlist.
- `remove` deletes the IP addresses and CIDRs listed in a text file from the allowlist.
- *`FILE_NAME`* is the name of a text file listing the IP addresses and CIDRs to be added to or removed from the allowlist. Each entry in the file must be separated by a newline character. Use the uploadFile command to upload this file to the environment. Each line in the file must be an IPv4 address or CIDR in the following format:

  ```
  xxx.xxx.xxx.xxx
  xxx.xxx.xxx.xxx/n
  ```

  > **Note:**
  >
  > – Only IPv4 IP addresses are supported.
  >
  > – Use CIDR format, rather than individual IP addresses, to specify a continuous range of IP addresses.
  >
  > – To disable an allowlist to permit access from any IP address, use the getIPAllowlist command to write all existing IP addresses and CIDRs to a file. Upload the file to the environment, and then run this command with the `remove` option as shown in this example:
  >
  > ```
  > epmAutomate getIPAllowlist > myRemoveList.txt
  > epmAutomate uploadFile myRemoveList.txt
  > epmAutomate setIPAllowlist remove myRemoveList.txt
  > ```

Examples

- Adding some IP addresses and CIDRs to an allowlist:

  ```
  epmAutomate setIPAllowlist add myAddList.txt
  ```

- Removing some IP addresses from an allowlist:

  ```
  epmAutomate setIPAllowlist remove myRemoveList1.txt
  ```

setManualDataAccess

Specifies whether Oracle is permitted to manually access the relational and Oracle Essbase databases of an environment in emergency situations when an environment is unresponsive and customer has not yet provided a service request to investigate and make the environment available.

In emergency situation, Oracle resorts to an internal process whereby a high-level development executive, after an independent verification process, permits manual access to the relational and Essbase databases. You use this command to stop Oracle from accessing these databases without your explicit approval. Additionally, you have the option to prohibit Oracle from manually accessing the relational and Essbase databases in emergency situations even if a service request is open.

The setting you specify using this command takes immediate effect.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setManualDataAccess Allow|Revoke [disableEmergencyAccess=true|false]`, where `disableEmergencyAccess`, optionally, specifies whether you want to prohibit all manual access to the relational and Essbase databases. Setting this property value to `true` stops Oracle from manually accessing these databases even if a service request is open. Default is `false`.
Oracle does not recommend setting `disableEmergencyAccess=true` because Oracle cannot access the relational and Essbase databases if access is required to troubleshoot and fix a down environment. If the environment is down, you will not be able to issue this command to allow Oracle to manually access these databases.

Examples

- Revoke the permission that was granted to manually access the relational and Essbase databases in emergency situations without explicit approval:
  ```
  epmautomate setManualDataAccess revoke
  ```
- Allow manual access to relational and Essbase databases during emergencies:
  ```
  epmautomate setManualDataAccess allow
  ```
- Prohibit manual access to the relational and Essbase databases even if a service request is open:
  ```
  epmautomate setManualDataAccess revoke disableEmergencyAccess=true
  ```

setPeriodStatus

Sets a specific status to a period.

Applies to

Account Reconciliation

Required Roles

Service Administrator, Power User, User, Viewer
Users with Power User, User, and Viewer predefined roles may require additional application roles.

Usage

`epmautomate setPeriodStatus` *`PERIOD STATUS`* where:

- *`PERIOD`* is the name of a period
- *`STATUS`* is `OPEN`, `CLOSED` or `LOCKED`

Example

```
epmautomate setPeriodStatus "January 2015" OPEN
```

setRestrictedDataAccess

Configures the Oracle Enterprise Performance Management Cloud environment to prevent Service Administrators from consenting to submit an application snapshot to Oracle while using the Provide Feedback utility.

If you have a policy to not allow Oracle to access your data, use this command to manage data sharing while submitting feedback to Oracle. Setting restricted data access for your environment disables the Submit Application Snapshot option that is displayed by the Provide Feedback utility.

Applies to

Account Reconciliation, Oracle Enterprise Data Management Cloud, Enterprise Profitability and Cost Management, Financial Consolidation and Close, FreeForm, Planning, Planning Modules, Profitability and Cost Management, Narrative Reporting, Sales Planning, Strategic Workforce Planning, and Tax Reporting.

Required Roles

Service Administrator

Usage

```
epmautomate setRestrictedDataAccess true|false
```
To view the current status of data access restrictions, use getRestrictedDataAccess.

Examples

- Prevent Service Administrators from consenting to submit an application snapshot:
  ```
  epmautomate setRestrictedDataAccess true
  ```
- Allow Service Administrators to consent to submit an application snapshot:

```
epmautomate setRestrictedDataAccess false
```

setSubstVars

Creates or updates substitution variables at application or cube level.

You cannot use this command to set multiple values and/or functions for substitution variables.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setSubstVars` *`CUBE_NAME SUBSTVAR=VALUE [SUBSTVAR=VALUE]`* where:

- *`CUBE_NAME`* is the cube (for example, Plan1, Plan2) for which the substitution variable is created or updated. Use `All` instead of a cube name to set or update substitution variable at the application level.
- *`SUBSTVAR`* is the name of the substitution variable for which a value is to be set or updated.
- *`VALUE`* is the new substitution variable value.

Examples

- Create or update one substitution variable at the application level: `epmautomate setSubstVars ALL CurYear=2015 CurPeriod=Jan`
- Create or update substitution variables at cube level: `epmautomate setSubstVars Plan2 CurYear=2013 CurPeriod=Jan`

setVirusScanOnFileUploads

Enables the OCI (Gen 2) environment to scan files for viruses before they are uploaded to Oracle Enterprise Performance Management Cloud.

All OCI (Gen 2) environments are protected using an anti-virus program. This command provides additional security by allowing you to enable virus scan on upload files. Scanning files before they are uploaded prevents the possibility of uploading viruses to the environment.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate setVirusScanOnFileUploads true|false`
By default, virus scan is not enabled (set to `false)`. When this value is set to `true`, EPM Cloud scans all upload files. If a file is virus infected, it is not uploaded to the environment.

Example

- Enable virus scan: `epmautomate setVirusScanOnFileUploads true`
- Disable virus scan: `epmautomate setVirusScanOnFileUploads false`

simulateConcurrentUsage

Executes different concurrent operations on an environment by simulating users.

This command can be used to validate the performance of the environment to verify that the response time is acceptable when the service is under the load during specific operations run by a specific number of users. For example, this command can be used to measure performance if 50 users simultaneously open a form using different POVs. It allows the self-service load testing of environments.

This command performs the simulation by executing the specified operations for a given number of users and iterations. It runs multiple iterations to calculate the minimum time, the maximum time and the average time for a particular operation. It supports these operations to perform concurrent usage load testing:

- Open forms
- Save forms
- Run business rules
- Run data rules
- Open ad hoc grids
- Execute Reports
- Execute Books

> **Note:**
>
> This command does not support Financial Reporting reports and books; it only supports books and reports belonging to Reports (formerly Management Reports).

> **Caution:**
>
> This command executes the specified operations on the current environment and may, depending on the operation, update the data in the environment. Run this command on test environments. Running this command on production environments is not advised.

This command accepts a ZIP file, that must already have been uploaded to the inbox of the environment, as input. The ZIP file contains one `requirement.csv` file and the input files that support the use cases included in `requirement.csv`. Optionally, the ZIP file may contain a

`userVarMemberMapping.csv` file to provide user variable member mapping, an `options.xml` file to provide Oracle Smart View for Office options for some use cases, and a `users.csv` file to provide the user names and passwords of the already existing users to use rather than creating new users. The command then simulates the use cases and creates a report that may be emailed to one or more recipients.

Usage Scenario 1: Acceptance testing of application performance for 50 users simultaneously opening a form.

Solution:

1. Create `requirement.csv` with entries similar to the following, assuming that you want to open a form named `Exchange Rates` stored in `Library/Global Assumption/` folder:

   ```
   # Type of Operation,Artifact Name,Number of Users,Input File,Additional
   Info
   Open Form, Library/Global Assumption/Exchange Rates,50,open_form_input.csv,
   ```

2. Create `open_form_input.csv` using the format specified in Open Form Input File. You will have one entry in this file, which will be used 50 times. If you want to open the same form with different POVs, you will have as many entries as the number of POVs you want to use.
3. Create `userVarMemberMapping.csv` using the format specified in Creating the UserVarMemberMapping.csv File if user variable member mapping needs to be set.
4. Export Smart View options into `options.xml`, if Smart View options need to be used. See Creating the options.xml File for details.
5. Create a ZIP file containing the files from the preceding steps and upload it to the inbox.
6. Run the `simulateConcurrentUsage` command using the ZIP file from the preceding step as the input file.

Usage Scenario 2: Simulating performance for seasonal usage increase, for example, at the end of the financial year. Assumption: 100 users save a form with a lag time of six seconds between each user.

Solution:

1. Create `requirement.csv` with entries similar to the following, assuming that you want to save a form named `Accessories Revenue` stored in `Library/Dashboards/` folder:

   ```
   # Type of Operation,Artifact Name,Number of Users,Input File,Additional
   Info
   Save Form, Library/Dashboards/Accessories Revenue,100,save_form_input.csv,
   ```

2. Create `save_form_input.csv` using the format specified in Save Form Input File.
3. Create `userVarMemberMapping.csv` using the format specified in Creating the UserVarMemberMapping.csv File if user variable member mapping needs to be set.
4. Export Smart View options into `options.xml`, if Smart View options need to be used. See Creating the options.xml File for details.
5. Create a ZIP file containing the files from the preceding steps and upload it to the inbox.
6. Run the `simulateConcurrentUsage` command using the ZIP file from the preceding step as the input file, and these property values `iteration=1` and `lagTime=6`.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator. You require Identity Domain Administrator role also to use testModes 0, 1, and 2.

Usage

`epmautomate simulateConcurrentUsage` *`INPUT_FILE.zip`* `[iterations=`*`COUNT`*`] [notificationEmails="EMAIL_ADDRESS"] [testMode=0|1|2|3|4] [lagTime=`*`LAG_TIME`*`]`
where:

- *`INPUT_FILE.zip`* is the name of a ZIP file that identifies your use cases. Use the uploadFile command (example command syntax `epmautomate uploadFile "C:/uploads/`*`INPUT_FILE.zip`*`" inbox`) to upload this file to the inbox before running this command. This ZIP file must contain these files:
  - A use case CSV file named `requirement.csv`. Each line of this CSV file identifies the type of operation to perform, artifact name, number of concurrent users, input file specifying the details of the operation, and additional information related to each use case. See Creating the requirement.csv File.
  - The input files that contains details of each operation. See these topics:
    * Open Form Input File
    * Save Form Input File
    * Run Business Rule Input File
    * Run Data Rule Input File
    * Ad Hoc Grid Input File
    * Execute Book Input File
    * Execute Report Input File
  - Optionally, add `userVarMemberMapping.csv` to the input ZIP file to provide user variable member mapping. See Creating the UserVarMemberMapping.csv File.
  - Optionally, add `options.xml` to the input ZIP file to use Smart View options. See Creating the options.xml File.
  - Optionally, add `users.csv` to the input ZIP file to provide the user names and passwords of existing users. See Creating the users.csv File.
- `iterations` is a positive number indicating the number of times each use case identified in `requirement.csv` is to be run to measure response time. If not specified, the operation is run only once.
- `notificationEmails`, optionally, indicates the email addresses to which the results of this commands are to be emailed. If specifying more than one email addresses, use a semicolon to separate them. Also enclose the list of address in double quotation marks. If not specified, the results are mailed to the user who initiated the command. For detailed information on this report, see Sample Simulate Concurrent Usage Report.
- `[testMode]`, optionally, specifies the concurrent usage simulation mode. Default is 0. Acceptable values are:

  - `0`: The default simulation mode, which adds simulated users to the environment and assigns them Service Administrator role, runs the simulation, and then deletes the simulated users. This mode is useful if you want to run the test only one time.
    The simulated users have these properties:

    First Name: testuser1, testuser2, and so on

    Last Name: testuser1, testuser2, and so on.

    Email Address: testuser1@discard.oracle.com, testuser2@discard.oracle.com, and so on

    Username: testuser1, testuser2, and so on
  - `1`: Adds simulated users to the environment and assigns them the Service Administrator role. Does not run the simulation or delete the simulated users.
    After using this mode, run the command with mode 3 to run the simulation as many times as needed . At the end, run the command with mode 2 to delete the simulated users.
  - `2`: Deletes simulated users. Does not create users or run the simulation.
  - `3`: Runs the simulation using already existing simulated users without adding or deleting users.
  - `4`: Uses the users defined in the `users.csv` file included in the input ZIP file to run the command. See Creating the users.csv File. This mode does not create users for simulation. Instead, it uses existing users.

  If you want to run the concurrent usage one time only, use `testMode=0`. To run a series of tests:

  - First, run the command using `testMode=1` to add the simulated users and assign them the Service Administrator role.
  - Then, run the command using `testMode=3` to run the simulation as many times as needed.
  - Finally, run the command using `testMode=2` to delete the simulated users.

- `[lagTime]`, optionally, specifies the number of seconds (5 seconds or more) that the command should wait between the execution of each use case in `requirement.csv`. Default is 5 seconds. Do not use negative numbers (for example -1), fractions (for example, 1/2), and decimal values.
  After initiating the execution of a use case in `requirement.csv` by one user, the command waits for the number of seconds specified by this parameter to initiate the execution of the use case by the next user. Because user activities are not usually initiated simultaneously, setting this parameter helps to create a more realistic simulation of load on an environment.

Example

```
epmautomate simulateConcurrentUsage test_simulation.zip iterations=5
notificationEmails="jane.doe@example.com;john.doe@example.com;example@example.
com" lagTime=6
```

skipUpdate

Requests that Oracle skip the applying of monthly updates to an environment for a maximum of three consecutive cycles or removes all skip update requests that were previously made using this command so that the environment is updated to the main code line.

You can use this command also to list the skip update requests currently specified for an environment. Skip update status of the environment is included in the Activity Report (in Operational metrics) generated after you use this command to skip updates to an environment. See Operational Metrics in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.

Weekly and emergency patches for the current month, if any, will continue to be applied to the environment. No updates will be made for the months for which the upgrade delay is requested.

You cannot skip updates for an environment that is on a one-off patch. Additionally, you cannot skip monthly updates that are more than three months apart from the update that the environment is currently on. For example, if the environment is currently on 23.12, you can skip 24.01, 24.02, and 24.03, but not 24.04. For detailed information on how update delays work, see Requesting Upgrade Delay for Production Environments in *Oracle Enterprise Performance Management Cloud Operations Guide*.

> **Note:**
>
> If you skip the update for only one of your environments (for example, you skip update on the production environment but not on the test environment) for three months, your environments will be three versions apart. You may not be able to migrate snapshots across these environments in such a scenario.
> For example, assume that your test and production environments are currently on 23.12, and that you skip the updates for versions 24.01, 24.02, and 24.03 for the production environment only. When version 24.03 becomes available, your test environment will be on version 24.03 while the production environment will still be on version 23.12. In this case, migration between your test and production environments is not supported.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate skipUpdate add|remove|list [version=`*`UPDATE_NUMBER`* `comment="`*`COMMENT`*`"]`
where:

- `add` sets a skip update request for a specific monthly update. You must specify these parameters:
  - `version`: the monthly update that should be skipped. You can skip one, two or three of the upcoming three monthly updates. For example, if the environment is on the 23.12 monthly update, you can skip the 24.01, 24.02, 24.03 or any of these updates. To skip three monthly updates, you run the command thrice, each time specifying one specific update to skip using, for example, `version=24.01`, then `version=24.02`, and then `version=24.03`. In this scenario, the environment will be updated to the main code line in the 24.04 monthly cycle.

If there is a gap in the monthly cycle for which skip update is requested and the current monthly cycle, Oracle will update the environment as required and then skip the updates in the specified monthly cycle. For example, the environment is on the 23.12 monthly update and you specify a skip update for versions 24.02 and 24.03. In this case, the environment will be updated in 24.01; 24.02 and 24.03 updates will be skipped. The environment will be updated to the main code line in 24.04.

  - `comment`: text describing why an update skip is required. Comments must be enclosed in double quotation marks.

- `remove` removes all skip update requests specified for the environment so that it can be updated to the main code line during the next daily maintenance. If you have more than one skip update requests on an environment, this command removes them all.
- `list` to display skip update requests (the login ID of the user who made the skip update request, comment, version for which updates are to be skipped, and the date when the request was made) currently set for the environment as shown in the following graphic:

```
c:\Oracle\EPM Automate\bin>epmautomate skipupdate add version=24.01 comment="Some Comment"
skipupdate completed successfully


c:\Oracle\EPM Automate\bin>epmautomate skipupdate add version=24.02 comment="Some Comment"
skipupdate completed successfully


c:\Oracle\EPM Automate\bin>epmautomate skipupdate add version=24.03 comment="Some Comment"
skipupdate completed successfully


c:\Oracle\EPM Automate\bin>epmautomate skipupdate list
skipupdate completed successfully

1] User:  example@example.com | Version: 24.03 | Comments: Some Comment | Timestamp: 2023-11-15T19:17:09Z
2] User:  example@example.com | Version: 24.02 | Comments: Some Comment | Timestamp: 2023-11-15T19:17:01Z
3] User:  example@example.com | Version: 24.01 | Comments: Some Comment | Timestamp: 2023-11-15T19:16:51Z

c:\Oracle\EPM Automate\bin>epmautomate skipupdate remove
skipupdate completed successfully
```

Examples

- Request a skip update: `epmautomate skipUpdate add version=24.01 comment="We are in the process of closing the quarter"`
- View skip update details: `epmautomate skipUpdate list`
- Remove all skip update requests: `epmautomate skipUpdate remove`

snapshotCompareReport

Compares two snapshots and creates the Snapshot Compare Report identifying the differences in calculation rules and rulesets, and data forms included in the snapshots. You can use this report for troubleshooting issues, such as:

- Recent performance deterioration in an environment. You can compare the previous snapshot with the current snapshot to check on the differences that may have caused the performance deterioration.
- You see difference in behavior or performance between two environments that you expect to have identical functional behavior or performance. In this case, you can compare the snapshots of the two environments to understand the differences between them.
- You suspect that some rules, or forms have disappeared from an environment. Use this report to compare the artifacts that used to exist and the current artifacts.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Enterprise Profitability and Cost Management, Strategic Workforce Planning, and Sales Planning.

Required Roles

Service Administrator

Usage

`epmAutomate snapshotCompareReport` *`SOURCE_SNAPSHOT TARGET_SNAPSHOT`* `[reportName=`*`REPORT_NAME`*`.html]` where:

- *`SOURCE_SNAPSHOT`* is the name of the snapshot to which the comparison is to be made. The report contains data on the differences in the rules, forms, dimensions, and dimension members in this snapshot.
- *`TARGET_SNAPSHOT`* is the name of the snapshot you want to compare.

> **Note:**
>
> – Snapshot names may be specified with or without the .ZIP extension.
>
> – Both snapshots must be available in the environment. Use the uploadFile, copyFromObjectStorage, or the copySnapshotFromInstance command to upload them to the environment.

- *`REPORT_NAME`*, optionally, is the name of the report file. Default report name is `SnapshotCompare.html`.
  Use the downloadFile command to download the report.

Examples

- `epmAutomate snapshotCompareReport "Artifact Snapshot" Backup_22-09-08.zip reportName=Snapshot_Diffs.html`
- `epmAutomate snapshotCompareReport backup_snapshot_22-Aug-08.zip backup_Snapshot_22-Sep-08.zip reportName=Sep_22_snapshot_compare_report.html`

sortMember

Sorts members of Entity, Account, Scenario, and Versions dimensions and of custom dimensions.

This command is useful for sorting dimension members after loading members into the application.

> **Note:**
>
> You cannot use this command to sort members of Period, Years, and Currency dimensions.

Applies to

Planning, Planning Modules, FreeForm, Enterprise Profitability and Cost Management, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate sortMember` *`Member`* `[type=children|descendants] [order=ascending|descending]` where:

- *`Member`* is the name of the parent member whose descendants or children are to be sorted.
- `type`, optionally, specifies the members to be sorted. Acceptable values are:
  - `descendants` sorts all sub-members (children and descendants) of the parent member that you specify as the value of `Member`
  - `children`, the default value, sorts only members in the level immediately below the parent member that you specify as the value of `Member`.
- `order`, optionally, identifies a sort order. Acceptable values are:
  - `ascending`; this is the default sort order.
  - `descending`

Examples

- Sort the children of Entity dimension in ascending order: `epmautomate sortMember Entity`
- Sorts all sub-members of the Entity dimension in descending order: `epmautomate sortMember Entity type=descendants order=descending`

unassignRole

Removes a role currently assigned to the users, including the user who runs this command, whose login IDs are included in the ANSI or UTF-8 encoded CSV file that is used with this command. You can use this command to remove the assignment of a predefined role or an application role.

The CSV file format is as follows:

```
User Login
jane.doe@example.com
jdoe
```

Use the uploadFile command to upload the file to the environment.

> **Note:**
>
> Use double quotation marks to enclose role names that contain the space character.

When the command execution finishes, EPM Automate prints information about each failed entry to the console. Review this information to understand why the command execution failed for some entries in the CSV file.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

To remove predefined role assignments:

- Classic environments: Identity Domain Administrator and any predefined role (Service Administrator, Power User, User or Viewer)
- OCI environments: Service Administrator, or Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

To remove application role assignments: Service Administrator or Access Control - Manage

Usage

`epmautomate unassignRole` *`FILE_NAME ROLE`* where:

- *`FILE_NAME`* is the name of a CSV file containing the login IDs of the users whose role assignment is to be revoked. Specify the CSV extension in lower case.
  User Login values are not case-sensitive. For example, `jane.doe@example.com` is treated as being identical to `Jane.Doe@Example.com` or any variation in its case.
- `ROLE` identifies one of the following. Role names are not case-sensitive.
  - If you are removing the assignment of users to predefined roles, `ROLE` should identify a predefined role applicable to the service. See Understanding Predefined Roles in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*.
  - If you are removing the assignment of users to application roles, `ROLE` should identify a role belonging to the application in the current environment. Application roles are listed in the **Roles** tab of Access Control. For a description of application roles for each business process, see these topics in *Administering Access Control for Oracle Enterprise Performance Management Cloud*:
    - Account Reconciliation
    - Enterprise Profitability and Cost Management
    - Planning, FreeForm, Financial Consolidation and Close, and Tax Reporting
    - Profitability and Cost Management
    - Oracle Enterprise Data Management
    - Narrative Reporting

Examples

- Unassign users from a predefined identity domain role:
  ```
  epmautomate unassignRole remove_roles.csv "Service Administrator"
  ```
- Unassign users from an application role:
  ```
  epmautomate unassignRole example_file.csv "Task List Access Manager"
  ```

updateUsers

Modifies attributes such as email, first name, and last name of Oracle Enterprise Performance Management Cloud users in an identity domain using the new values identified in an ANSI or UTF-8 encoded Comma Separated Value (CSV) file that was uploaded to the environment.

You use the uploadFile command to upload files to an environment. All columns in the CSV file are mandatory; you must provide a valid entry in each column. This command validates each definition for these mandatory values and displays error message that identifies each missing or invalid value. The input 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
```

If the User Login value in the CSV file matches an account that exists in the identity domain, the command modifies the user account to match the values in the input file. Because user accounts are common to all environments that an identity domain supports, updated user information is available to all the environments that share the identity domain. Predefined and application-specific roles assigned to the user are not affected by this command

> **Note:**
>
> - You cannot use this command to modify User Login values.
> - You are not permitted to modify your own account attributes.
> - Input file containing multi-byte characters must use UTF-8 character encoding. Using ANSI encoding causes issues in how user information is displayed in My Services screens.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Strategic Workforce Planning, and Sales Planning.

Required Roles

Identity Domain Administrator and any predefined role (Service Administrator, Power User, User, or Viewer)

Usage

`epmautomate updateUsers` *`FILE_NAME`* where *`FILE_NAME`* is the name of a CSV file containing the user information to modify.

Example

```
epmautomate updateUsers update_user_info.csv
```

upgrade

Automatically downloads and silently installs the newest version of EPM Automate.

After you run the login command to initiate a session, EPM Automate identifies the current installed version. If the installed version is not the newest available, EPM Automate informs you that a newer version is available.

> **Note:**
>
> EPM Automate deployed by a Windows administrator can be upgraded only if the logged in user is a Windows administrator.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User, User, Viewer

Usage

```
epmautomate upgrade
```

Example

```
epmautomate upgrade
```

uploadFile

Uploads a file from the local computer to the service. Use this command to upload files containing data, metadata, rule definitions, dimension definitions, mapped transactions, templates, and backup snapshots.

This command does not overwrite existing files in the environment. EPM Automate displays an error if the name of the file being uploaded is identical to that of a file in the upload location.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator, Power User assigned to the Migration Administrator application role

Usage

`epmautomate uploadFile "FILE_NAME" [UPLOAD_LOCATION]` where:

- *FILE_NAME* is the name of the file, including absolute path if the file is not in the directory from which you are running EPM Automate.
- *UPLOAD_LOCATION* is, optionally, the Oracle Enterprise Performance Management Cloud location to which you want to upload the file. Do not specify an upload location if you want to upload files to the default upload location. For detailed information, see Default File Locations. Supported values include:
  - `inbox` to upload files into the inbox. Excepting Profitability and Cost Management, EPM Cloud business processes look in this location for files to process.
  - `profitinbox` to upload files to be processed by Profitability and Cost Management.
  - `to_be_imported` to upload a Narrative Reporting snapshot that is to be imported during the next daily maintenance of the environment.
  - `inbox/`*`directory_name`* to upload files to a directory within the inbox for processing by Data Management.
  - `outbox` to upload files to the outbox used by business processes other than Profitability and Cost Management.
  - `profitoutbox` to upload files to the outbox used by Profitability and Cost Management.

Examples

- Upload a snapshot into the default location:
  ```
  epmautomate uploadFile "C:/snapshots/backup_snapshot.zip"
  ```
- Upload a file into the Data Management inbox:
  ```
  epmautomate uploadFile "C:/pbcsdata/quarterlydata.csv" inbox
  ```
- Upload a file into a folder in the inbox (for Data Management):
  ```
  epmautomate uploadFile "C:/fdmee_data/data.zip" inbox/dm_folder
  ```
- Upload a file into the profitinbox (Profitability and Cost Management):
  ```
  epmautomate uploadFile "C:/profitability_data/data.zip" profitinbox
  ```
- Upload Narrative Reporting snapshot from `C:\temp` directory into the to_be_imported location:
  ```
  epmautomate uploadFile "C:\temp\EPRCS_Backup.tar.gz" to_be_imported
  ```

userAuditReport

Generates a user audit report (.CSV file) and stores it in the default download location.

The User Audit Report contains information on the users who signed into an environment over a specified period of time (maximum last 120 days). It lists the user login ID, the IP address of the computer from which the user logged in, and the date and time (for example, July 28, 2022 18:43:21 UTC) at which the user accessed the environment.

> **Note:**
>
> The User Audit Report lists only one login entry for a user who logged into an Oracle Enterprise Performance Management Cloud environment multiple times within a span of five minutes.

	A	B	C
1	Type Of Report	User Audit Report	
2	Report Generated Date	6/1/2020	
3			
4	User Name	IP Address	Access Date and Time
5			
6	jdoe@example.com	111.22.23.6	June 1, 2020 22:28:00 UTC
7	jane.doe@example.com	111.22.23.6	June 1, 2020 21:40:56 UTC

Use the downloadFile to download the generated report to your computer.

Applies to

Planning, Planning Modules, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator

Usage

`epmautomate userAuditReport` *`FROM_DATE TO_DATE REPORT_NAME`* where:

- *`FROM_DATE`* indicates the start date (in `YYYY-MM-DD` format) of the period for which the audit report is to be generated
- *`TO_DATE`* indicates the end date (in `YYYY-MM-DD` format) of the period for which the audit report is to be generated
- *`REPORT_NAME`* is the name of the report file

> **Note:**
>
> This report can be generated only for the last 120 days.

Example

```
epmautomate userAuditReport 2016-10-15 2016-12-15 myAuditReport.CSV
```

userGroupReport

Generates a report (CSV file) that lists the groups to which users are assigned in Access Control and stores it in the default download location.

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 Login,First Name,Last Name,Email,Direct,Group
jdoe, John, Doe, jdoe@example.com, Yes, test1
jdoe, John, Doe, jdoe@example.com, No, test2
```

Use the downloadFile to download the generated report to your computer.

Applies to

Planning, Planning Modules,, FreeForm, Financial Consolidation and Close, Tax Reporting, Account Reconciliation, Profitability and Cost Management, Enterprise Profitability and Cost Management, Oracle Enterprise Data Management Cloud, Narrative Reporting, Sales Planning, and Strategic Workforce Planning.

Required Roles

Service Administrator or Access Control - Manage

Usage

`epmautomate userGroupReport` *`REPORT_NAME`* where *`REPORT_NAME`* is the name of the report file.

Example

```
epmautomate userGroupReport UsrGrpReport.CSV
```

validateConsolidationMetadata

Validates the metadata of the environment to ensure an error free database refresh and consolidation.

After importing metadata using the importMetadata command, run this command to validate the metadata to ensure an error free database refresh when you run the refreshCube command. If your consolidation metadata is not correct, your consolidation may also fail.

This command displays 0 (zero) or a count of the number of validation errors on the console from which it is run. It writes validation errors to a CSV file, which you can use to correct the metadata errors. Use the downloadFile command to download the resulting CSV file to a local server.

Applies to

Financial Consolidation and Close

Required Roles

Service Administrator

Usage

`epmautomate validateConsolidationMetadata` *`LOG_FILE_NAME`* where `LOG_FILE_NAME` identifies the name of the file that will contain information on the errors identified by this command.

Examples

```
epmautomate validateConsolidationMetadata validation_error.csv
```

validateModel

Validates an Enterprise Profitability and Cost Management model and writes validation output to a file.

Applies to

Enterprise Profitability and Cost Management

Required Roles

Service Administrator

Usage

`epmautomate validateModel "modelName" FILE_NAME.txt [messageType=All|Warning|Error]` where:

- `modelName` is the name of the Enterprise Profitability and Cost Management model to be validated. This value must be enclosed in double quotation marks.
- *`FILE_NAME`* is a unique name for a text file to which EPM Automate should write model validation output. This file, created in the outbox, can be downloaded using the downloadFile command.
- `messageType`, optionally, is the status of information to include in the model validation output. Possible parameter values are:
  - `All`, which writes both errors and warnings to the validation output file.
  - `Error`, which records only errors in the validation output file. This is the default value.
  - `Warning`, which records only model validation warnings in the validation output file.

Example

```
epmautomate validateModel "10 Actuals Allocation Process" results.txt
messageType=All
```

Exit Codes

EPM Automate returns an exit code and message to indicate the status of the operation. Exit codes are grouped under five code numbers; each code may indicate many error conditions. Review the accompanying message to identify the specific condition that caused the error.

Additionally, EPM Automate creates a log file (*`COMMANDNAME_TIMESTAMP.log`*, for example, `uploadfile_16_11_2016_11_27_10.log`) for each failed command execution. Log files are created on the computer from which you run EPM Automate.

Exit Code 1 Errors

`Command failed to execute` EPM Automate uses this exit code to display messages related to HTTP status code 200 and 400. These codes are returned by the REST APIs that EPM Automate uses.

`Insufficient privileges to perform the operation` This error is displayed if the user whose credentials are used to sign into the service does not have sufficient privileges to perform the operation you attempted.

Sign in with an account that has sufficient privileges to perform the operation. Generally, only Service Administrators are allowed to perform operations in the service.

`Resource does not exist` This error is displayed if the file or snapshot that you want to delete or download does not exist in the service.

Use the `listfiles` command to verify the file name and its location.

`Invalid snapshot` ***`SNAPSHOT`*** This error is displayed when the service is unable to validate the snapshot that you specified for the export or import operation.

Verify that you are using a valid snapshot.

`Internal server error. Unable to delete file:` ***`FILE_NAME`*** **`Please issue "Provide Feedback" with details`** This error is displayed if the file or snapshot could not be deleted from the service due to a server error.

Report this issue to Oracle using the Feedback command or the Provide Feedback feature.

`Invalid file:` ***`FILE_NAME`*** This error is displayed if the file or snapshot that you want to delete or download does not exist in the service or if the file name is not in the required format.

Use the `listfiles` command to verify the file name and its location.

`Recreate is running for a long time. Please contact support` This error is displayed if the re-create operation that you initiated is not completed within one hour.

Report this issue to Oracle using the feedback command or the Provide Feedback feature.

`Reset service is running for a long time. Please contact support` This error is displayed if the reset service operation that you initiated is not completed within one hour.

Report this issue to Oracle using the feedback command or the Provide Feedback feature.

`Cannot perform operation. Another instance is in progress. Please try after some time` This error is displayed if you try to execute the `copysnapshotfrominstance` command when another instance of the command is active.

Wait for the `copysnapshotfrominstance` command to finish before attempting to run the command again.

`Cannot perform operation. Another maintenance script is in progress. Please try after some time` This error is displayed if you attempt to execute the `copysnapshotfrominstance`, `recreate` or `resetservice` command when daily maintenance or service reset process is running.

Rerun the operation after the maintenance or reset process finishes.

`Login to source instance failed:` ***`SOURCE_URL`*** This error is displayed if EPM Automate is unable to sign in to the source environment to initiate the `copysnapshotfrominstance` command

Verify that the credentials, identity domain and URL that is used to access the source environment are valid.

`Internal server error. Copy snapshot from source instance failed. Please issue "Provide Feedback" with details` This error is displayed when EPM

Automate encounters an unexpected problem while running the `copysnapshotfrominstance` command.

Report this issue to Oracle using the `feedback` command or the Provide Feedback feature.

`Internal server error. Please issue "Provide Feedback" with details` This error is displayed to indicate many internal server conditions that require corrective actions by Oracle.

Report this issue to Oracle using the `feedback` command or the Provide Feedback feature.

`Snapshot SNAPSHOT_NAME already exists. Please delete the snapshot and rerun the command` This error is displayed when you download or upload a snapshot into a location where another snapshot with identical name is present.

Delete or remove the existing snapshot and then retry the command.

`Error in extracting the snapshot. Please retry with a proper snapshot` This error is displayed if EPM Automate is unable to extract snapshot contents when running the `importsnapshot` command.

Verify that the snapshot is valid.

`Internal server error. Unable to open file for write. Please issue "Provide Feedback" with details` This error is displayed if errors cause the creating or updating of CSV files, for example while generating the Audit Report.

Report this issue to Oracle using the `feedback` command or the Provide Feedback feature.

`No matching records found, please select a different date range` This error is displayed if you run the `userauditreport` command to generate the Audit Report for a date range for which audit data is not available.

Specify a valid date range and then rerun the `userauditreport` command. Note that the service maintains audit history for the last 365 days only.

`File with same name exists: FILE_NAME, please choose a different filename` This error is displayed if a report with the Audit Report name you specified exists in the service.

Delete the existing file from the service or specify a different file name and then rerun the `userauditreport` command.

`Operation failed with status $1. Please issue "Provide Feedback"` This message indicates an internal server error that cause the reset service or re-create service process to fail.

Report this issue to Oracle using the `feedback` command or the Provide Feedback feature.

`EPMCSS-20643: Failed to add users. File FILE_NAME`.csv is not found. Please provide a valid file name This error is displayed if the specified CSV file containing information on users to add is not available in the Inbox.

Use the `listfiles` command to verify the file name and its location. If the file is not in the inbox, use the uploadFile command to upload the file.

`EPMCSS-20644: Failed to remove users. File FILE_NAME.csv is not found. Please provide a valid file name` This error is displayed if the specified CSV file containing information on users to delete is not available in the Inbox.

Use the `listfiles` command to verify the file name and its location. If the file is not in the Inbox, use the uploadFile command to upload the file.

`20645: Failed to assign role for users. Invalid role name role. Please provide a valid role name` This error is displayed if the role specified in the CSV file is not supported.

Verify that the role name specified in the file is `Service Administrator`, `Power User`, `User`, or `Viewer`.

Use the `listfiles` command to verify the file name and its location. If the file is not in the Inbox, use the uploadFile command to upload the file.

Exit Code 6 Errors

`Service Unavailable` Service is not available because of HTTP Error 404.

Verify service availability by accessing the service from a browser on the computer from which you are running EPM Automate. If the service is down for any reason, wait a while and try again or contact Oracle support.

`Read/Write timeout` This error is displayed if the client socket times out (socket time out is 15 minutes) during any read/write operation due to slow network or firewall issues.

Rerun the failed command when network through put is high. If the failure is due to firewall settings, contact your Network Administrator.

Exit Code 7 Errors

EPM Automate displays this error if it is unable to execute a command. The error message; for example, `Invalid command`, specifies why the error occurred.
`Unable to open password file FILE_NAME` Invalid encrypted password file, for example, *`PASSWORD_FILE`*`.EPW`. EPM Automate did not find the file in the location that you specified or the file is not in the required format.

Verify the file name and path. If the file cannot be parsed because of invalid format, use the encryptcommand to re-create the file.

`Unable to parse password file FILE_NAME` EPM Automate was unable to parse the encrypted password file because of invalid format or because it has been corrupted.

Use the encrypt command to re-create the file.

`Unable to connect to URL. Root cause MESSAGE` This error is displayed if a connection cannot be established because of a bad URL. The message that is displayed as the root cause details the underlying failure resulting from using an incorrect URL.

- Verify that you are using a valid URL
- If your proxy setting requires you to authentication with the proxy server to connect to the internet, specify a proxy server user name, domain, and password (or use an encrypted password file containing the proxy server password) to sign in. Contact your Network Administrator if you need assistance.

`Unable to connect to URL Unsupported Protocol` The login command failed because the URL specified uses an unsupported protocol. Accompanying error message identifies the unsupported protocol.

Ensure that the URL that you are using with the login command uses the secure protocol (HTTPS).

`Session is not authenticated. Please execute the "login" command before executing any other command` You attempted to run a command before establishing a session using the `login` command.

Run the login command to establish a secure connection to the environment before attempting to execute other commands.

`Invalid parameter` This message indicates a usage error in a command caused by Incorrect sequence of command parameters or the absence of some required command parameter values.

Review and correct command parameters and the sequence in which they are specified.

`COMMAND_NAME`* `command is not supported by` *`SERVICE_TYPE` EPM Automate was not able to run the command against the environment to which you are connected because the business process does not support the command.

See Command Reference for lists of commands that are supported by each business process.

`File does not exist in location:` *`PATH`* EPM Automate was unable to find the file that you want to process, for example, using the `upload` or `replay` command.

Ensure that the file name and path are accurate.

`Unable to open file for read:` *`PATH`* EPM Automate was unable to read from the specified file.

Ensure that the file is in the required format. Verify that the user running EPM Automate has read access to the file.

`Unable to open file for write:` *`PATH`* EPM Automate was unable to write to the specified file.

Ensure that the file is not locked by another process. Verify that the user running EPM Automate has write access to the file.

`Invalid command` EPM Automate encountered an unsupported command.

Verify that EPM Automate supports the command; also ensure that the command name is spelled correctly.

`Invalid date format` The tool encountered an invalid date format.

Specify the report generation dates in a supported date format.

`FROMDATE` *`DATE`* `cannot be greater than TODATE` *`DATE`* EPM Automate encountered a to date that is earlier than the from date.

Ensure that the `to date` in a specified date range is a later date than the `from date`.

`Exceeded maximum number of feedbacks (6) for a day` This error is displayed when you exceed the number of feedback that you can submit using the `feedback` command.

`File with the same name already exists in the download path` *`PATH`*. Please delete the file and rerun the command This error is displayed when you attempt to download a file into a location that already has a file that matches the name of the file being downloaded.

Delete, rename or move the existing file and then rerun the command.

`File is empty:` *`PATH`* This error is displayed if the replay file does not have any content.

Make sure that the replay file (CSV file) lists the credentials (user name and password) and the name of the HAR files that are to be used to run the `replay` command.

`Unable to encrypt the password as localhost cannot be resolved. Ensure that hostnames are mapped properly to IP addresses` This error is displayed if EPM Automate is unable to resolve the localhost definition to a MAC address because the hosts file on your computer contains a server name instead of `localhost` for the address `127.0.0.1`.

Ensure that the hosts file specify `localhost` as the server name for `127.0.0.1`

`Snapshot Name is invalid` This error is displayed if you do not specify the name of the snapshot to be renamed.

Specify the name of a snapshot available in the environment.

`New Snapshot Name is invalid` This error is displayed if you do not specify a new name for the snapshot.

specify the new name for the snapshot.

`Invalid snapshot name: {0}. Characters \\/*?"<>| are not allowed` This error is displayed if the snapshot name contains special characters such as space, \ (backslash), / (slash), * (asterisk), ? (question mark), " (quotation mark), < (less than), and > (greater than).

Specify a new snapshot name that does not contain these special characters.

`Unable to rename snapshot : {0}. There could be another process accessing it. Please try after sometime` This error is displayed if EPM Automate cannot get an exclusive lock on the snapshot because it is in use by another process.

Wait until the current operation that is using the snapshot finishes, and then retry.

`Snapshot {0} already exist. Please delete the snapshot and re-run the command` This error is displayed if the new snapshot name is identical to that of an existing snapshot in the environment.

Use a different snapshot name or delete the existing snapshot using the `deletefile` command.

Exit Code 9 Errors

`Invalid credentials` This error is displayed when the user name or password used with the login command is incorrect.

Specify valid credentials for the environment to which you are attempting to connect.

`Authentication failed while executing command. Please retry` This error is displayed when basic authentication fails during execution of a command other than login. This error may also occur for HTTP calls when a command execution is retried (up to three times).

Exit Code 11 Errors

Internal server error. Due to manual reset service, your Oracle EPM Cloud Service environment is currently unavailable. This error is displayed if EPM Automate commands are run when a reset of the environment is in progress.

`Internal server error MESSAGE` This error is displayed if EPM Automate encounters unknown exceptions that are not related to HTTP connections. Includes server errors 503 and 500.

`Unable to connect to URL: MESSAGE` This error is displayed when the server is unavailable. Error message indicates the exception that caused the command to fail.

If the server is unavailable, contact Oracle Support. If the message indicates issues with the URL, verify that the URL that you are using is valid.

3

Command Execution Sample Scenarios

EPM Automate may be used to automate many common Oracle Enterprise Performance Management Cloud administrative tasks.

- Sample Scenarios for All Services
- Sample Scenarios for Planning, Consolidation, Tax Reporting, and Enterprise Profitability and Cost Management
- Sample Scenarios for Account Reconciliation
- Sample Scenarios for Profitability and Cost Management
- Sample Scenarios for Oracle Enterprise Data Management Cloud

About Copying Sample Scripts

Do not copy sample scripts from the PDF version of this document. To avoid line breaks and footer information that will render scripts unusable, Oracle recommends that you copy the sample scripts from the HTML version of *Working with EPM Automate for Oracle Enterprise Performance Management Cloud*.

Sample Scenarios for All Services

These scenarios depict a typical sequence of commands that can be used to perform specific operations in Oracle Enterprise Performance Management Cloud environments.

Related Topics

- Back up Application Snapshot to a Computer
  This scenario explains how to automate the process of backing up the snapshot created during daily service maintenance to a local computer.
- Inform Users of Daily Maintenance Completion
  The daily maintenance of Oracle Enterprise Performance Management Cloud environments usually takes a much shorter time than the one hour earmarked for it.
- Copying a Snapshot to or from Oracle Object Storage
- Create Users and Assign Them to Predefined Roles
  Use the scripts in this section to create users and assign them to predefined roles in the identity domain.
- Count the Number of Licensed Users (Users Assigned to Roles)
  Use the script in this section to generate the Role Assignment Report to count the number of users for an environment.
- Create Audit Reports of Users Assigned to Roles
  Use the scripts in this section to automate the process of creating an audit report for users assigned to predefine roles in an environment and, optionally, email it to a recipient.
- Create Role Assignment and Revocation Audit Report
  Use the PowerShell script in this section to automate the process of creating an audit report that details role assignment and role revocation in an environment.

- Mask Access Logs and Activity Report to Comply with Privacy Laws
  Use the scripts in this section to automate the process of masking information in the Activity Report or Access Logs to comply with privacy laws and to, optionally, email the report to a recipient.
- Automate Activity Report Downloads to a Local Computer
  Use the script in this section to automate the downloading of Activity Reports from an environment to a local computer.
- Download Access Logs from an Environment
  Use the script in this section to automate the process of downloading access logs from an environment to a local computer.
- Automate the Cloning of Environments
  Use the script in this section to automate the cloning of environments.
- Clone from Primary to Standby Environment Daily After Daily Maintenance is Complete on the Primary Environment
  To keep the standby environment up to date with the primary environment, use these scripts to clone Oracle Enterprise Performance Management Cloud primary environment to the standby environment soon after the daily maintenance is complete on the primary environment.
- Remove Unnecessary Files from an Environment
  Use these scripts to remove unnecessary files from an environment.
- Find and Download Files from an Environment
  Use the sample script in this section to automate the process of downloading one or more files from an Oracle Enterprise Performance Management Cloud environment using a text string as a wildcard.
- Recreate an Old EPM Cloud Environment for Audits
  Use the script in this section to create a self-service solution to maintain an up-to-date library of snapshots for your Oracle Enterprise Performance Management Cloud environment. You require an environment dedicated for the purpose of upgrading and maintaining a library of up-to-date snapshots.
- Automate Database Access Audit and Compliance
  Use the PowerShell and Bash Shell scripts in this section to leverage EPM Automate commands to collect audit and compliance data around manual database access.
- Replicate Users and Predefined Role Assignments
  The scripts in this section helps you migrate users and predefined role assignments of an environment to another.
- Create a Quarterly EPM Cloud Upgrade Cadence
  Use these scripts to create a self-service solution to skip updates so that Oracle Enterprise Performance Management Cloud environments are updated on a quarterly basis with a two-week test cycle. In this case, the production environments are updated two weeks after test environments.
- Create a Quarterly EPM Cloud Upgrade Cadence with Six Week Test Cycles
  Use the script in this section to create a self-service solution to skip updates so that Oracle Enterprise Performance Management Cloud environments are updated on a quarterly basis with a six-week test cycle. In this case, the production environments are updated six weeks after the test environments.

Back up Application Snapshot to a Computer

This scenario explains how to automate the process of backing up the snapshot created during daily service maintenance to a local computer.

- Downloads the application snapshot (`Artifact Snapshot`) that was created during the maintenance window
- Renames the downloaded snapshot by appending time stamp
- Maintains 10 backups by deleting the oldest backup, if needed

> **Note:**
>
> - This script cannot be used to backup Narrative Reporting
> - If you repurpose this script for your use, modify the values of runtime parameters (`url`, `user`, `password`, and `NumberOfBackups` as needed.

See Automating Script Execution for information on scheduling the script using Windows Task Scheduler.

Windows Sample Script

Create a batch (`.bat`) or shell (`.sh`) file containing script similar to the following to automate snapshot downloads.

```
@echo off
rem Sample script to download and maintain 10 maintenance backups
rem Update the following  parameters

SET url=https://example.oraclecloud.com
SET user=ServiceAdmin
SET password=Example.epw
SET SnapshotName="Artifact Snapshot"
SET NumberOfBackups=10

rem EPM Automate commands
call epmautomate login %user% %password% %url%
     IF %ERRORLEVEL% NEQ 0 goto :ERROR
       call epmautomate downloadfile %SnapshotName%
     IF %ERRORLEVEL% NEQ 0 goto :ERROR
       call epmautomate logout
     IF %ERRORLEVEL% NEQ 0 goto :ERROR

rem Rename downloaded Artifact Snapshot, keep the last 10 backups
Set Timestamp=%date:~4,2%_%date:~7,2%_%date:~10,2%%
Set Second=%time:~0,2%%time:~3,2%
ren %SnapshotName%.zip %SnapshotName%_%Timestamp%_%Second%.zip

SET Count=0
FOR %%A IN (%SnapshotName%*.*) DO SET /A Count += 1
IF %Count% gtr %NumberOfBackups% FOR %%A IN (%SnapshotName%*.*) DO del "%%A"
&& GOTO EOF
:EOF

echo Scheduled Task Completed successfully
exit /b %errorlevel%
:ERROR
```

```
echo Failed with error #%errorlevel%.
exit /b %errorlevel%
```

Linux/UNIX Sample Script

Create a shell (`.sh`) file containing script similar to the following to automate the snapshot downloads. If your password contains special characters, see Handling Special Characters.

```
#!/bin/sh
# Sample script to download and maintain 10 maintenance backups
# Update the following seven parameters

url=https://example.oraclecloud.com
user=serviceAdmin
password=/home/user1/epmautomate/bin/example.epw
snapshotname="Artifact Snapshot"
numberofbackups=10
epmautomatescript=/home/user1/epmautomate/bin/epmautomate.sh
javahome=/home/user1/jdk1.8.0_191/

export JAVA_HOME=${javahome}

printResult()
    {
    op="$1"
    opoutput="$2"
    returncode="$3"

    if [ "${returncode}" -ne 0 ]
    then
        echo "Command failed. Error code: ${returncode}. ${opoutput}"
    else
        echo "${opoutput}"
    fi
}

processCommand()
{
    op="$1"
    date=`date`

    echo "Running ${epmautomatescript} ${op}"
    operationoutput=`eval "$epmautomatescript $op"`
    printResult "$op" "$operationoutput" "$?"
}

op="login ${user} ${password} ${url}"
processCommand "${op}"

op="downloadfile \"${snapshotname}\""
processCommand "${op}"

op="logout"
processCommand "${op}"

# Renames the downloaded artifacts, keeps the last 10 backups
```

```
timestamp=`date +%m_%d_%Y_%I%M`
mv "${snapshotname}.zip" "${snapshotname}_${timestamp}.zip"

((numberofbackups+=1))
ls -tp ${snapshotname}*.zip | grep -v '/$' | tail -n +${numberofbackups} |
xargs -d '\n' -r rm --
```

Inform Users of Daily Maintenance Completion

The daily maintenance of Oracle Enterprise Performance Management Cloud environments usually takes a much shorter time than the one hour earmarked for it.

The actual daily maintenance duration of the environment is recorded as the value of the "Daily Maintenance Duration in Minutes" metric in the "Operations Metrics" section of the Activity Report. If you do not want to wait for the whole hour before using the environment, use a custom version of this script to inform users that the daily maintenance is complete so that they can resume activities.

Windows Script

Create `daily_maintenance_completed.ps1` by copying the following PowerShell script. See Running the Script for information on updating the script for your use.

```
# Daily Maintenance Completed Notification script
#
# Update the following parameters
# -------------------------------
$emailaddresses=user1@oracle.com,user2@oracle.com
# -------------------------------

$username=$args[0]
$password=$args[1]
$url=$args[2]

if ($($args.count) -ne 3) {
    echo "Usage: ./daily_maintenance_completed.ps1 <USERNAME> <PASSWORD>
<URL>"
    exit 1
}

$amw_time=""

function getDailyMaintenanceStartTime {
    $amwstring=$(epmautomate.bat getDailyMaintenanceStartTime)
    $elements=$amwstring.split(' ')
    $amwtime=$elements[0]
    return $amwtime
}

function goToSleep ($amw_time){
    $current_mdy=Get-Date -AsUTC -UFormat "%m/%d/%Y"
    $current_date_time=Get-Date -AsUTC -UFormat "%m/%d/%Y %H:%M:%S"
    $current_epoch=Get-Date -Date $current_date_time -UFormat "%s"
    $target_date_time=[DateTime]"${current_mdy} ${amw_time}"
    $target_epoch=Get-Date -Date $target_date_time -UFormat "%s"
    $sleep_seconds=$target_epoch - $current_epoch
```

```
    # Today's AMW start time has already passed, so add 24 hours to
sleep_seconds
    if ($sleep_seconds -lt 0) {
        $sleep_seconds=$sleep_seconds + 86400
    }

    $sleep_ts=New-TimeSpan -Seconds ${sleep_seconds}
    $sleep_hms="${sleep_ts}" -replace '^\d+?\.'

    echo "Current time is ${current_date_time}. Sleeping for ${sleep_hms},
until daily maintenance start time of ${amw_time}."
    Start-Sleep -Seconds $sleep_seconds
}

function attemptLogin {
    $serverdown=$False
    while ($true) {
        epmautomate.bat login ${username} ${password} ${url}
        if ($?) { # login succeeded
        if ($serverdown) { # server has been brought down
                echo "Daily maintenance processing has completed ..."
                break
        } else { # server has not yet been brought down
                echo "Daily maintenance processing has not yet started.
Sleeping for 2 minutes before the next check ..."
                Start-Sleep -Seconds 120
            }
    } else { # login failed
        if ($serverdown) { # server has been brought down
                echo "Waiting for daily maintenance processing to complete.
Sleeping for 2 minutes before the next check ..."
                Start-Sleep -Seconds 120
        } else { # server has not yet been brought down
                echo "Daily maintenance processing is now beginning. Sleeping
for 2 minutes before the next check ..."
                Start-Sleep -Seconds 120
            $serverdown=$True
        }
        }
    }
}

function sendNotification {
    $servername=$url.split("/")[2];
    $subject="Daily maintenance processing has completed"
    $formattedmessage="Daily maintenance processing has completed for server $
{servername}"
    $emailaddresses=${emailaddresses}.replace(',',';')

    echo "Mailing report"
    epmautomate.bat sendmail "${emailaddresses}" "${subject}" Body="$
{formattedmessage}"
}

echo "Beginning daily maintenance completion notification script."
```

```
echo "Logging into server ..."
epmautomate.bat login ${username} ${password} ${url}
$amwtime=getDailyMaintenanceStartTime
goToSleep ($amwtime)
attemptLogin
sendNotification
echo "Logging out of server ..."
epmautomate.bat logout
echo "Script processing has completed."
```

Linux/UNIX Script

Create `daily_maintenance_completed.sh` by copying the following script. See Running the Script for information on updating the script for your use.

```
#!/bin/bash
# Update the following parameters
# -------------------------------
epmautomatescript="LOCATION_EPM_AUTOMATE_EXECUTABLE"
javahome="LOCATION_JAVA_HOME"
emailaddresses=EMAIL_ADDRESS_1,EMAIL_ADDRESS_2,EMAIL_ADDRESS_N
# -------------------------------

username="$1"
password="$2"
url="$3"

export JAVA_HOME=${javahome}

if [ "$#" -ne 3 ]; then
    echo "Usage: ./daily_maintenance_completed.sh <USERNAME> <PASSWORD> <URL>"
    exit 1
fi

amw_time=""

getDailyMaintenanceStartTime() {
    amw_time=$(${epmautomatescript} getDailyMaintenanceStartTime | cut -d' ' -
f1)
}

goToSleep() {
    current_mdy=$(date -u +%m/%d/%Y)
    current_date_time=$(date -u)
    current_epoch=$(date +%s)
    target_epoch=$(date -d "${current_mdy} ${amw_time}" +%s)
    sleep_seconds=$(($target_epoch - $current_epoch))

    # Today's AMW start time has already passed, so add 24 hours to
sleep_seconds
    if [[ ${sleep_seconds} -lt 0 ]]
    then
        sleep_seconds=$((sleep_seconds + 86400))
    fi

    sleep_hms=$(date -d@${sleep_seconds} -u +%H:%M:%S)
```

```
    echo "Current time is ${current_date_time}. Sleeping for ${sleep_hms},
until daily maintenance start time of ${amw_time}."
    sleep $sleep_seconds
}

attemptLogin() {
    local serverdown=1
    while true
    do
        ${epmautomatescript} login ${username} ${password} ${url}
        if [[ $? -eq 0 ]] # login succeeded
        then
            if [[ ${serverdown} -eq 0 ]] # server has been brought down
            then
                echo "Daily maintenance processing has completed"
                break
            else # server has not yet been brought down
                echo "Daily maintenance processing has not yet started.
Sleeping for 2 minutes before the next check ..."
                sleep 120
            fi
        else # login failed
            if [[ ${serverdown} -eq 0 ]] # server has been brought down
            then
                echo "Waiting for daily maintenance processing to complete.
Sleeping for 2 minutes before the next check ..."
                sleep 120
            else # server has not yet been brought down
                echo "Daily maintenance processing is now beginning. Sleeping
for 2 minutes before the next check ..."
                sleep 120
                serverdown=0
            fi
        fi
    done
}

sendNotification()
{
    local servername=$(echo "${url}" | cut -d '/' -f3- | rev | cut -d':' -f2-
| rev)
    local subject="Daily maintenance processing has completed"
    local formattedmessage="Daily maintenance processing has completed for
server ${servername}"
    local emailaddresses=$(echo ${emailaddresses} | sed "s/,/;/g")

    echo "Mailing report"
    ${epmautomatescript} sendmail "${emailaddresses}" "${subject}" Body="$
{formattedmessage}"
}

echo "Beginning daily maintenance completion notification script."
echo "Logging into server ..."
${epmautomatescript} login ${username} ${password} ${url}
getDailyMaintenanceStartTime
```

```
goToSleep
attemptLogin
sendNotification
echo "Logging out of server ..."
${epmautomatescript} logout
echo "Script processing has completed."
```

Server-Side Groovy Script

Create `daily_maintenance_completed` Groovy script by copying the following code. See Running the Script for information on updating the script for your use.

```
// Daily Maintenance Completed Notification script

// Update the following parameters
// -------------------------------
String username="USERNAME"
String password="PASSWORD"
String url="URL OF THE ENVIRONMENT"
String emailaddresses="EMAIL_ADDRESS_1,EMAIL_ADDRESS_2,EMAIL_ADDRESS_N"
// -------------------------------

def LogMessage(String message) {
    def date = new Date()
    def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss")
    println('[' + sdf.format(date) + '] ' + message);
}

def LogOperationStatus(EpmAutomateStatus opstatus) {
    def returncode = opstatus.getStatus()
    if (returncode != 0){
        LogMessage(opstatus.getOutput())
    }
    LogMessage('return code: ' + returncode)
}

def getDailyMaintenanceStartTime(EpmAutomate automate) {
    LogMessage("Operation: getDailyMaintenanceStartTime")
    EpmAutomateStatus amwtimestatus =
automate.execute('getDailyMaintenanceStartTime')
    LogOperationStatus(amwtimestatus)
    def amwstring=(amwtimestatus.getOutput())
    def elements=amwstring.split(' ')
    def amwtime=elements[0]
    return amwtime
}

def goToSleep(String amw_time){
    def date = new Date()
    def current_mdy = new SimpleDateFormat("MM/dd/yyyy")
    def current_date_time = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss")
    float current_epoch = date.getTime() / 1000
    def pattern = "MM/dd/yyyy HH:mm:ss"
    def input = current_mdy.format(date) + " " + amw_time + ":00"
    def target_date_time = Date.parse(pattern, input)
    float target_epoch = target_date_time.getTime() / 1000
```

```
    int sleep_seconds = Math.round(target_epoch - current_epoch)

    //Today's AMW start time has already passed, so add 24 hours to
sleep_seconds
    if (sleep_seconds < 0) {
        sleep_seconds = sleep_seconds + 86400
    }

    def sleep_milliseconds = sleep_seconds * 1000
    LogMessage("Current time is " + current_date_time.format(date) + ".
Sleeping until daily maintenance start time of " + amw_time + ":00.")
    sleep(sleep_milliseconds)
}

def attemptLogin(EpmAutomate automate, String username, String password,
String url) {
    def serverdown=1
    while (true) {
        LogMessage("Operation: login " + username + " " + password + " " +
url)
        EpmAutomateStatus status =
automate.execute('login',username,password,url)
        def returncode = status.getStatus()
        if (returncode == 0) {
            if (serverdown == 0){
                LogMessage("Daily maintenance processing has completed ...")
                break
            } else {
                LogMessage("Daily maintenance processing has not yet started.
Sleeping for 2 minutes before the next check ...")
                sleep(120000)
            }
        } else {
            if (serverdown == 0){
                LogMessage("Waiting for daily maintenance processing to
complete. Sleeping for 2 minutes before the next check ...")
                sleep(120000)
            } else {
                LogMessage("Daily maintenance processing is now beginning.
Sleeping for 2 minutes before the next check ...")
                sleep(120000)
                serverdown=0
            }
        }
    }
}

def sendNotification(EpmAutomate automate, String url, String emailaddresses)
{
    def servername=url.tokenize("/")[-1];
    def subject="Daily maintenance processing has completed"
    def formattedmessage="Daily maintenance processing has completed for
server " + servername
    def emailaddressesformatted = emailaddresses.replaceAll(',',';')

    LogMessage("Operation: sendmail " + emailaddressesformatted + " " +
```

```
subject + " Body=" + formattedmessage)
    EpmAutomateStatus status =
automate.execute('sendmail',emailaddressesformatted,subject,'Body=' +
formattedmessage)
    LogOperationStatus(status)
}

LogMessage("Beginning daily maintenance completion notification script.")

EpmAutomate automate = getEpmAutomate()

LogMessage("Operation: login " + username + " " + password + " " + url)
EpmAutomateStatus status = automate.execute('login',username,password,url)
LogOperationStatus(status)

String amwtime = getDailyMaintenanceStartTime(automate)
goToSleep (amwtime)
attemptLogin(automate,username,password,url)
sendNotification(automate,url,emailaddresses)

LogMessage("Operation: logout ")
status = automate.execute('logout')
LogOperationStatus(status)

LogMessage ("Script processing has completed.")
```

Running the Script

Windows and Linux/UNIX

1. Create `daily_maintenance_completed.ps1` or `daily_maintenance_completed.sh` by copying the script from a preceding section.
2. Update script:
   - **Windows:** Update the value of `emailaddresses` with a comma separated list of email addresses that should be notified when the daily maintenance is complete.
   - **Linux/UNIX:** Update these variables:
     - `epmautomatescript` with the location of the EPM Automate executable. Example: `epmautomatescript="/home/utils/EPMAutomate/bin/epmautomate.sh"`
     - `javahome` with the directory where the JDK used by EPM Automate is installed. For example: `"/home/user1/jdk1.8.0_191"`
     - `emailaddresses` with a comma separated list of email addresses that should be notified when the daily maintenance is complete. For example: `jdoe@example.com,jane_doe@example.com`
3. In a Command Window or Console, navigate to the folder where the `daily_maintenance_completed` script is stored.
4. Run this command:
   - **Windows**: `./daily_maintenance_completed.ps1 *USERNAME PASSWORD URL*`
   - **Linux/UNIX**: `./daily_maintenance_completed.sh *USERNAME PASSWORD URL*`, where:
     - `USERNAME` is the user name of a Service Administrator
     - `PASSWORD` is the password of the Service Administrator

- `URL` is the URL of the EPM Cloud environment

Server-Side Groovy:

1. Create `daily_maintenance_completed.groovy` Groovy script by copying it from a preceding section.
2. Update these values.
   - `username` with the username of a Service Administrator.
   - `password` with the password of the Service Administrator
   - `url` with the URL of the EPM Cloud environment for which the daily maintenance completion notification needs to be made. For example: . Example: `https://testExample-idDomain.pbcs.us1.oraclecloud.com`
   - `emailaddresses` with a comma separated list of email addresses that should be notified when the daily maintenance is complete.
3. Use the Groovy screen in an EPM Cloud business process or automate the script execution using runBusinessRule. For more information see these information sources:
   - Running Commands without Installing EPM Automate
   - Using Groovy Rules in *Administering Planning*

Copying a Snapshot to or from Oracle Object Storage

This topic contains sample scripts to complete these tasks:

- Copy `Artifact Snapshot` (the maintenance snapshot) from Oracle Enterprise Performance Management Cloud to an Oracle Object Storage bucket and rename it by appending the date on which the snapshot was copied.
- Copy a backup snapshot from an Oracle Object Storage bucket to EPM Cloud.

The scripts in this section assume that you have already created a bucket in Oracle Object Storage to hold the snapshot. Before running these scripts, customize them for your use by updating these place holders:

Table 3-1 Parameters and Their Values

Place Holder	Expected Value
`JAVA_HOME`	Directory where the JDK used by EPM Automate is installed. **Example:** `./home/JDK/bin`
`epmautomateExe`	Directory where EPM Automate is installed. **Example:** `./home/utils/EPMAutomate/bin`
`cloudServiceUser`	User ID of an EPM Cloud Service Administrator. **Example:** `John.doe@example.com`
`cloudServicePassword`	Password of the Service Administrator or the location of the password file. If the password contains special characters, see Handling Special Characters. **Example:** `ex_PWD_213`
`cloudServiceUrl`	URL of the EPM Cloud environment from which `Artifact Snapshot` is to be copied. **Example:** `https//test-cloud-id_Dom.pbcs.us1.oraclecloud.com`

Table 3-1 (Cont.) Parameters and Their Values

Place Holder	Expected Value
objectStorageUser	User ID of a user in Oracle Object Storage. To copy a snapshot to Object Storage, this user must have write access for the bucket to which the snapshot is copied. To copy a snapshot from Object Storage, this user must have read access for the bucket from which the snapshot is copied. **Example:** jDoe
objectStoragePassword	Password of the objectStorageUser. **Example:** example_PWD
objectStorageBucketUrl	URL of the Oracle Object Storage bucket where the snapshot is to be copied. See these information sources for the URL format: • copyToObjectStorage • copyFromObjectStorage **Example:** https//swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/axaxnpcrorw5/bucket-20210301-1359
snapshot	Name of the snapshot that you want to copy from the Oracle Object Storage bucket. **Example:** Artifact Snapshot20210429.zip

Sample EPM Automate Script to Copy a Snapshot from EPM Cloud to Oracle Object Storage

Customize and run this script to rename and then copy Artifact Snapshot from EPM Cloud to an Oracle Object Storage bucket.

```
#!/bin/sh
export JAVA_HOME=<path_to_jdk>
epmautomateExe=<path_to_epmautomate_executable>
cloudServiceUser=<cloud_service _user>
cloudServicePassword=<cloud_service_password>
cloudServiceUrl=<cloud_service_url>
# User with write access to Object Storage bucket
objectStorageUser=<object_storage_user>
objectStoragePassword=<object_storage_password>
objectStorageBucketUrl=<object_storage_bucket>
currentDate=`date +'%Y%m%d'`
sourceSnapshot="Artifact Snapshot"
targetSnapshot="${sourceSnapshot} ${currentDate}"
$epmautomateExe login ${cloudServiceUser} ${cloudServicePassword} $
{cloudServiceUrl}
$epmautomateExe renamesnapshot "${sourceSnapshot}" "${targetSnapshot}"
$epmautomateExe copyToObjectStorage "${targetSnapshot}" ${objectStorageUser} $
{objectStoragePassword} "${objectStorageBucketUrl}/${targetSnapshot}"
$epmautomateExe logout
exit 0
```

Sample EPM Automate Script to Copy a Snapshot from Oracle Object Storage to EPM Cloud

Customize and run this script to copy `Artifact Snapshot` of a specific date from an Oracle Object Storage bucket to EPM Cloud.

```
#!/bin/sh
export JAVA_HOME=<path_to_jdk>
epmautomateExe=<path_to_epmautomate_executable>
cloudServiceUser=<cloud_service _user>
cloudServicePassword=<cloud_service_password>
cloudServiceUrl=<cloud_service_url>
# User with read access to Object Storage bucket
objectStorageUser=<object_storage_user>
objectStoragePassword=<object_storage_password>
objectStorageBucketUrl=<object_storage_bucket>
snapshot=<desired_snapshot>
$epmautomateExe login ${cloudServiceUser} ${cloudServicePassword} $
{cloudServiceUrl}
$epmautomateExe copyFromObjectStorage ${objectStorageUser} $
{objectStoragePassword} "${objectStorageBucketUrl}/${snapshot}"
$epmautomateExe logout
exit 0
```

Create Users and Assign Them to Predefined Roles

Use the scripts in this section to create users and assign them to predefined roles in the identity domain.

These scripts use EPM Automate commands to complete these activities:

- Sign in to the environment as a Service Administrator with the Identity Domain Administrator role.
- Export groups and membership information from the environment to regenerate a snapshot that you specify; for example, `example_snapshot.zip`. It is assumed that you previously exported Groups and Membership artifacts using Migration to create this snapshot.
- Download the snapshot (`example_snapshot.zip`) to a local directory.
- Delete the snapshot (`example_snapshot.zip`) from the environment.
- Sign out of the environment.
- Perform operations for which you added custom code. Such operations may include:
  - Extracting the contents of `example_snapshot.zip`
  - Appending new user information to `HSS-Shared Services\resource\External Directory\Users.csv` in First Name,Last Name,Email,User Login format. For detailed information, see Importing a Batch of User Accounts in *Getting Started with Oracle Cloud*.
  - Appending information about role assignments of new users (in First Name,Last Name,Email,User Login format) to appropriate roles file(s) in `HSS-Shared Services\resource\External Directory\Roles\`. For example, assignment to Service Administrator role should be appended to `<service_name> Service`

Administrator.csv while assignments to Viewer roe should be appended to HSS-Shared Services\resource\External Directory\Roles\<service_name> Viewer.csv. For detailed information, see Assigning One Role to Many Users in *Getting Started with Oracle Cloud*.

  - Recreating the snapshot with the updated files by zipping the HSS-Shared Services directory and its contents.

- Sign into the environment as a Service Administrator who also has Identity Domain Administrator role.
- Upload the modified example_snapshot.zip to the environment.
- Import example_snapshot.zip into the environment.
- Delete the uploaded example_snapshot.zip from the environment.
- Sign out.

Windows Sample Script

Create a file named createUsersAndAssignRoles.ps1 by copying the following script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$snapshotName="$($inputproperties.snapshotName)"
$userspassword="$($inputproperties.userspassword)"
$resetPassword="$($inputproperties.resetPassword)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate exportsnapshot ${snapshotName}
epmautomate downloadfile ${snapshotName}.zip
epmautomate deletefile ${snapshotName}.zip
epmautomate logout

# Add custom code to extract the contents of example_snapshot.zip
# Add custom code to append new user information to HSS-Shared
Services\resource\External Directory\Users.csv
# Optional: Add custom code to append role information to the appropriate
role file(s) in HSS-Shared Services\resource\External Directory\Roles\
# Add custom code to zip HSS-Shared Services and its contents as
example_snapshot.zip

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${snapshotName}.zip
epmautomate importsnapshot ${snapshotName} userPassword=${userspassword}
resetPassword=${resetPassword}
epmautomate deletefile ${snapshotName}.zip
epmautomate logout
```

Linux/UNIX Sample Script

Create a file named `createUsersAndAssignRoles.sh` by copying the following script. Store it in a local directory.

```
#!/bin/bash

. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} exportsnapshot "${snapshotName}"
${epmautomatescript} downloadfile "${snapshotName}.zip"
${epmautomatescript} deletefile "${snapshotName}.zip"
${epmautomatescript} logout

# Add custom code to extract the contents of example_snapshot.zip
# Add custom code to append new user information to HSS-Shared
Services\resource\External Directory\Users.csv
# Optional: Add custom code to append role information to the appropriate
role file(s) in HSS-Shared Services\resource\External Directory\Roles\
# Add custom code to zip HSS-Shared Services and its contents as
example_snapshot.zip

${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${snapshotName}.zip"
${epmautomatescript} importsnapshot "${snapshotName}" "userPassword=$
{userspassword}" "resetPassword=${resetPassword}"
${epmautomatescript} deletefile "${snapshotName}.zip"
${epmautomatescript} logout
```

Sample input.properties File

To run the `createUsersAndAssignRoles` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `createUsersAndAssignRoles.ps1` or `createUsersAndAssignRoles.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
snapshotName=SNAPSHOT_NAME
userspassword=TEMP_IDM_PASSWORD
resetPassword=true
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
snapshotName=SNAPSHOT_NAME
userspassword=TEMP_IDM_PASSWORD
resetPassword=true
```

Table 3-2 input.properties Parameters

Parameter	Description
javahome	JAVA_HOME location. For Linux/UNIX only.
epmautomatescript	Absolute path of EPM Automate executable (epmautomate.sh). For Linux/UNIX only.
username	User name of a Service Administrator, who also has the Identity Domain Administrator role.
password	Password of the Service Administrator or the name and location of the encrypted password file.
serviceURL	URL of the environment from which you want to generate the snapshot.
snapshotName	A name for the snapshot you want to generate. It is assumed that you previously exported Groups and Membership artifacts using Migration to create this snapshot.
userspassword	The initial password for new users.
resetPassword	Whether the new users must reset password on first login. Set this value to trueto force new users to change their password when they login for the first time.

Running the Script

1. Create createUsersAndAssignRoles.ps1 or createUsersAndAssignRoles.sh by copying the script from a preceding section.
2. Add custom code to perform these operations:
   - Extract the contents of the snapshot
   - Append new user information to HSS-Shared Services\resource\External Directory\Users.csv.
   - Optionally, append information about role assignments of new users (in First Name,Last Name,Email,User Login format) to appropriate roles file(s) in HSS-Shared Services\resource\External Directory\Roles\.
   - Recreate the snapshot with the updated files.
3. Create the input.properties file and save it in the directory where the createUsersAndAssignRoles script is located. Contents of this file differs depending on your operating system. See Sample input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
4. Launch the script.
   - **Windows PowerShell:** run createUsersAndAssignRoles.ps1.
   - **Linux/UNIX:** run ./createUsersAndAssignRoles.sh.

Count the Number of Licensed Users (Users Assigned to Roles)

Use the script in this section to generate the Role Assignment Report to count the number of users for an environment.

Create provisionedUsersCount.bat by copying the following script.

> **Note:**
>
> - Input parameters to run `provisionedUsersCount.bat` are `username`, `password/password_file`, `service_url`, and `report_file_name`. For example, at the command prompt, enter a command similar to the following: `provisionedUsersCount.bat jdoe password.epw https://example.oraclecloud.com myRole_assign.CSV`
> - If the password contains special characters, see Handling Special Characters.

```
@echo off

set paramRequiredMessage=Syntax: provisionedUsersCount.bat USERNAME PASSWORD/
PASSWORD_FILE URL REPORT_FILE_NAME

if "%~1" == "" (
      echo User Name is missing.
    echo %paramRequiredMessage%
    exit /b 1
  )

if "%~2" == "" (
    echo Password or Password_File is missing.
    echo %paramRequiredMessage%
    exit /b 1
  )

if "%~3" == "" (
    echo URL is missing.
    echo %paramRequiredMessage%
    exit /b 1
  )

if "%~4" == "" (
    echo Role Assignment Report File Name is missing.
    echo %paramRequiredMessage%
    exit /b 1
  )

call epmautomate.bat login %~1 "%~2" %~3
REM call epmautomate.bat login %~1 "%~2" %~3

if %errorlevel% NEQ 0 exit /b 1
   call epmautomate.bat roleAssignmentReport "%5"
if %errorlevel% NEQ 0 exit /b 1
   call epmautomate.bat downloadFile "%5"
if %errorlevel% NEQ 0 exit /b 1

set filePath="%cd%\%4"

if exist %filePath% (
    SETLOCAL EnableDelayedExpansion
    set /a lineCount=0
```

```
    set /a userCount=0
    set userHeaderFound=false
    for /f "tokens=*" %%A in ( 'type %filePath%' ) do (
        set /a lineCount+=1
        set line=%%A

        REM Fetch username from role assignment information row
        if !userHeaderFound!==true (
            for /f "delims=," %%i in ("!line!") do (
                set userName=%%i
          )
        if NOT !userName! == "" (
          if !userCount! gtr 0 if NOT !userName! == !lastUserName! (
              set /a userCount+=1
          set users[!userCount!]=!userName!
                )
          if !userCount! == 0 (
                set /a userCount+=1
                set users[!userCount!]=!userName!
                )
      set lastUserName=!userName!
            )
        )

        REM Check for headers of Role Assignment Report
        if "!line!"=="User Login,First Name,Last Name,Email,Role,Granted
through Group" (
                set userHeaderFound=true
            )
    if "!line!"=="User Login,First Name,Last Name,Email,Roles,Granted
Through" (
        set userHeaderFound=true
         )
    )

    echo Number of Users: !userCount!
    for /l %%n in (1,1,!userCount!) do (
  REM echo !users[%%n]!
    )
    endlocal

) else (
      echo Invalid provisioning report file path - %filePath%.
)
```

Create Audit Reports of Users Assigned to Roles

Use the scripts in this section to automate the process of creating an audit report for users assigned to predefine roles in an environment and, optionally, email it to a recipient.

This audit report shows the users assigned to predefined roles or groups that changed since the last time the report was generated. To create a daily audit report, run this script on a daily basis.

Create `provisioningAuditReport.bat` by copying the following script. This wrapper batch script calls the PowerShell script `provisioningAuditReport.ps1`, the source code for which is provided later on in this scenario.

> **Note:**
>
> - Input parameters for running `provisioningAuditReport.bat` are: `username`, `password` or `password_file`, `service_url`, and `report_email_to_address` (optional, required only if you want to send the report to an email address).
> - If the password contains special characters, see Handling Special Characters.

```
@echo off
set paramRequiredMessage=Syntax: provisioningAuditReport.bat USERNAME
PASSWORD/PASSWORD_FILE URL [REPORT_EMAIL_TO_ADDRESS]

if "%~1" == "" (
  echo User Name is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~2" == "" (
  echo Password or Password_File is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~3" == "" (
  echo URL is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )

PowerShell.exe -File provisioningAuditReport.ps1 %*
```

`provisioningAuditReport.bat` calls `provisioningAuditReport.ps1`, which you create by copying the following script.

`provisioningAuditReport.ps1` creates the audit report. Place it in the same directory where `provisioningAuditReport.bat` is located.

```
$username=$args[0]
$password=$args[1]
$url=$args[2]
$reportemailtoaddress=$args[3]

$date=$(get-date -f dd_MM_yy_HH_mm_ss)
$datedefaultformat=$(get-date)
$logdir="./logs/"
$logfile="$logdir/epmautomate-provisionauditreport-" + $date + ".log"
$reportdir="./reports/"
$provisionreport="provreport-audittest-" + $date + ".csv"
$provisionreporttemp="./provreport-audittest-temp.csv"
```

```
$provisionreportunique="./provreport-audittest-unique.csv"
$provisionreportbaselineunique="./provreport-audittest-baseline-unique.csv"

function EchoAndLogMessage
{
  $message=$args[0]
  echo "$message"
  echo "$message" >> $logfile
}

function Init
{
  $logdirexists=Test-Path $logdir
  if (!($logdirexists)) {
    mkdir $logdir 2>&1 | out-null
    }

  $logfileexists=Test-Path $logfile
  if ($logfileexists) {
    rm $logfile 2>&1 | out-null
    }

  $reportdirexists=Test-Path $reportdir
  if (!($reportdirexists)) {
    mkdir $reportdir 2>&1 | out-null
    }
}

function PostProcess
{
  rm $provisionreporttemp
  mv -Force $provisionreportunique $provisionreportbaselineunique
}

function ProcessCommand
{
  $op=$args
  echo "EPM Automate operation: epmautomate.bat $op" >> $logfile
  epmautomate.bat $op >> $logfile 2>&1
  if ($LASTEXITCODE -ne 0) {
    echo "EPM Automate operation failed: epmautomate.bat $op. See $logfile
for details."
    exit
    }
}

function RunEpmAutomateCommands
{
  EchoAndLogMessage "Running EPM Automate commands to generate the
provisioning report."
  ProcessCommand login $username $password $url
  ProcessCommand provisionreport $provisionreport
  ProcessCommand downloadfile $provisionreport
  ProcessCommand deletefile $provisionreport
  ProcessCommand logout
}
```

```
function CreateProvisionReportTempFile
{
  # Loop through iteration csv file and parse
  Get-Content $provisionreport | ForEach-Object {
  $elements=$_.split(',')
  echo "$($elements[0]),$($elements[2])" >> $provisionreporttemp
  }
}

function CreateUniqueElementsFile
{
  gc $provisionreporttemp | sort | get-unique > $provisionreportunique
}

function CheckBaselineAndCreateAuditReport
{
  $provisionreportbaselineuniqueexists=Test-
Path $provisionreportbaselineunique
  if (!($provisionreportbaselineuniqueexists)) {
    EchoAndLogMessage "No existing provisioning report, so comparison with a
baseline is not possible. Audit report will be created at the next test run."
  } else {
    CreateAuditReport
    }
}

function EmailAuditReport
{
  $auditreport=$args[0]
  $elements=$auditreport.split('/')
  $auditreportname=$elements[2]

  if (${reportemailtoaddress} -match "@") {
    EchoAndLogMessage "Emailing audit report"
    ProcessCommand login $username $password $url
    ProcessCommand uploadFile $auditreport
    ProcessCommand sendMail $reportemailtoaddress "Provisionining Audit
Report" Body="Provisioning Audit Report is attached."
Attachments=$auditreportname
    ProcessCommand deleteFile $auditreportname
    ProcessCommand logout
  }
}

function CreateAuditReport
{
  $auditreport=$reportdir + "auditreport-"+ $date + ".txt"
  $additions = @()
  $deletions = @()

  EchoAndLogMessage "Comparing previous provisioning report with the current
report."
  $compare=compare-object (get-content $provisionreportunique) (get-
content $provisionreportbaselineunique)
```

```
  $compare | foreach  {
    if ($_.sideindicator -eq '<=')
     {
        $additions += $_.inputobject
     } elseif ($_.sideindicator -eq '=>') {
        $deletions += $_.inputobject
     }
  }

  echo "Provisioning Audit Report for $datedefaultformat" > $auditreport
  echo "-----------------------------------------------" >> $auditreport

  if ($additions.count -ne 0)
  {
    echo " "          >> $auditreport
    echo "Additions:" >> $auditreport
    foreach($element in $additions) { echo "$element" >> $auditreport }
    }

  if ($deletions.count -ne 0)
  {
    echo " "          >> $auditreport
    echo "Deletions:" >> $auditreport
    foreach($element in $deletions) { echo "$element" >> $auditreport }
  }

  if (($additions.count -eq 0) -and ($deletions.count -eq 0))
  {
    echo " "                                  >> $auditreport
    echo "No changes from last audit report." >> $auditreport
  }

  EchoAndLogMessage "Provisioning audit report has been
generated: $auditreport."
  EmailAuditReport $auditreport
}

Init
EchoAndLogMessage "Starting EPMAutomate provisioning audit reporting"
RunEpmAutomateCommands
CreateProvisionReportTempFile
CreateUniqueElementsFile
CheckBaselineAndCreateAuditReport
PostProcess
EchoAndLogMessage "EPMAutomate provisioning audit reporting completed"
```

Create Role Assignment and Revocation Audit Report

Use the PowerShell script in this section to automate the process of creating an audit report that details role assignment and role revocation in an environment.

Create `AuditReportRoleAssignment.bat` by copying the following script. This wrapper batch script calls the PowerShell script `AuditReportRoleAssignment.ps1`, the source code for which is provided later on in this scenario.

> **Note:**
>
> - Input parameters for running `AuditReportRoleAssignment.bat` are `username`, `password` or `password_file`, and `service_url`.
> - If the password contains special characters, see Handling Special Characters.

Script: AuditReportRoleAssignment.bat

```
@echo off
set paramRequiredMessage=Syntax: AuditReportRoleAssignment.bat USERNAME
PASSWORD/PASSWORD_FILE URL

if "%~1" == "" (
   echo User Name is missing.
   echo %paramRequiredMessage%
   exit /b 1
   )
if "%~2" == "" (
   echo Password or Password_File is missing.
   echo %paramRequiredMessage%
   exit /b 1
   )
if "%~3" == "" (
   echo URL is missing.
   echo %paramRequiredMessage%
   exit /b 1
   )

PowerShell.exe -File AuditReportRoleAssignment.ps1 %*
```

Script: AuditReportRoleAssignment.ps1

```
# EPM Automate Role Assignment Audit Report Script
$username=$args[0]
$password=$args[1]
$url=$args[2]

# Generic variables
$date=$(get-date -f dd_MM_yy_HH_mm_ss)
$datedefaultformat=$(get-date)
$logdir="./logs/"
$logfile="$logdir/epmautomate-provisionauditreport-" + $date + ".log"
$reportdir="./reports/"
$provisionreport="provreport-audittest-" + $date + ".csv"
$provisionreporttemp="./provreport-audittest-temp.csv"
$provisionreportunique="./provreport-audittest-unique.csv"
$provisionreportbaselineunique="./provreport-audittest-baseline-unique.csv"

function EchoAndLogMessage
{
    $message=$args[0]
    echo "$message"
```

```
    echo "$message" >> $logfile
}
function Init
{
    $logdirexists=Test-Path $logdir
    if (!($logdirexists)) {
        mkdir $logdir 2>&1 | out-null
    }
    $logfileexists=Test-Path $logfile
    if ($logfileexists) {
        rm $logfile 2>&1 | out-null
    }
    $reportdirexists=Test-Path $reportdir
    if (!($reportdirexists)) {
        mkdir $reportdir 2>&1 | out-null
    }
}

function PostProcess
{
    rm $provisionreporttemp
    mv -Force $provisionreportunique $provisionreportbaselineunique
}

function ProcessCommand
{
    $op=$args
    echo "EPM Automate operation: epmautomate.bat $op" >> $logfile
    epmautomate.bat $op >> $logfile 2>&1
    if ($LASTEXITCODE -ne 0) {
        echo "EPM Automate operation failed: epmautomate.bat $op.
See $logfile for details."
        exit
    }
}

function RunEpmAutomateCommands
{
    EchoAndLogMessage "Running EPM Automate commands to generate the audit
report."
    ProcessCommand login $username $password $url
    ProcessCommand provisionreport $provisionreport
    ProcessCommand downloadfile $provisionreport
    ProcessCommand deletefile $provisionreport
    ProcessCommand logout
}
function CreateProvisionReportTempFile
{
    # Loop through iteration csv file and parse
    Get-Content $provisionreport | ForEach-Object {
        $elements=$_.split(',')
        echo "$($elements[0]),$($elements[2])" >> $provisionreporttemp
    }
}

function CreateUniqueElementsFile
```

```
{
    gc $provisionreporttemp | sort | get-unique > $provisionreportunique
}

function CheckBaselineAndCreateAuditReport
{
    $provisionreportbaselineuniqueexists=Test-
Path $provisionreportbaselineunique
    if (!($provisionreportbaselineuniqueexists)) {
        EchoAndLogMessage "Could not find a baseline audit report to compare
with. Audit report will be created next time you run test."
    } else {
        CreateAuditReport
    }
}

function CreateAuditReport
{
    $auditreport=$reportdir + "auditreport-"+ $date + ".txt"
    $additions = @()
    $deletions = @()
    EchoAndLogMessage "Comparing previous audit report with the current one."
    $compare=compare-object (get-content $provisionreportunique) (get-
content $provisionreportbaselineunique)
    $compare | foreach  {
        if ($_.sideindicator -eq '<=')
        {
            $additions += $_.inputobject
        } elseif ($_.sideindicator -eq '=>') {
            $deletions += $_.inputobject
        }
    }
    echo "Provisioning Audit Report for $datedefaultformat" > $auditreport
    echo "-----------------------------------------------" >> $auditreport
    if ($additions.count -ne 0)
    {
        echo " "          >> $auditreport
        echo "Additions:" >> $auditreport
        foreach($element in $additions) { echo "$element" >> $auditreport }
    }
    if ($deletions.count -ne 0)
    {
        echo " "          >> $auditreport
        echo "Deletions:" >> $auditreport
        foreach($element in $deletions) { echo "$element" >> $auditreport }
    }
    if (($additions.count -eq 0) -and ($deletions.count -eq 0))
    {
        echo " "                                  >> $auditreport
        echo "No changes from last audit report." >> $auditreport
    }
    EchoAndLogMessage "Role audit report generated: $auditreport."
}

Init
EchoAndLogMessage "Starting EPMAutomate role audit report generation"
```

```
RunEpmAutomateCommands
CreateProvisionReportTempFile
CreateUniqueElementsFile
CheckBaselineAndCreateAuditReport
PostProcess
EchoAndLogMessage "EPMAutomate role audit report completed"
```

Mask Access Logs and Activity Report to Comply with Privacy Laws

Use the scripts in this section to automate the process of masking information in the Activity Report or Access Logs to comply with privacy laws and to, optionally, email the report to a recipient.

Because of the stringent privacy laws of some countries, the information available in the Activity Reports and Access Logs may have to be hidden from Service Administrators to protect privacy of users.

You use `anonymizeData.bat` to mask information in the Activity Report or Access Logs to comply with privacy laws and to, optionally, email it. To mask information, schedule this script or a variation there of using Windows scheduler so that it runs everyday soon after the daily maintenance process for each environment is complete.

Use these information sources:

- Using Activity Reports and Access Logs to Monitor Usage in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*
- Automating Script Execution

You manually create `anonymizeData.bat` by copying the Windows script provided in the following procedure and schedule it using Windows scheduler. You may create and run similar platform-appropriate scripts if you are not using Windows for scheduling.

`anonymizeData.bat` is a wrapper script, which executes the `anonymizeData.ps1` script, which you create and update as explained in the following procedure.

If the password you use contains special characters, see Handling Special Characters

1. Create a batch (`BAT`) file named `anonymizeData.bat` containing the following script and save it in a convenient location, for example, `C:\automate_scripts`.

```
@echo off
set paramRequiredMessage=Syntax: anonymizeData.bat USERNAME PASSWORD/
PASSWORD_FILE URL [EMAIL_TO_ADDRESS]

if "%~1" == "" (
  echo User Name is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~2" == "" (
  echo Password or Password_File is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~3" == "" (
  echo URL is missing.
  echo %paramRequiredMessage%
```

```
  exit /b 1
  )

PowerShell.exe -File anonymizeData.ps1 %*
```

2. Create a PowerShell script (`PS1`) file named `anonymizeData.ps1` containing the following script and save it in a convenient location, for example, `C:\automate_scripts`.

```
# Anonymize data script

$username=$args[0]
$password=$args[1]
$url=$args[2]
$emailtoaddress=$args[3]

# Generic variables
$date=$(get-date -f dd_MM_yy_HH_mm_ss)
$datedefaultformat=$(get-date)
$logdir="./logs/"
$logfile="$logdir/anonymize-data-" + $date + ".log"
$filelist="filelist.txt"

function LogMessage
{
  $message=$args[0]

  echo "$message" >> $logfile
}

function EchoAndLogMessage
{
  $message=$args[0]

  echo "$message"
  echo "$message" >> $logfile
}
function Init
{
   $logdirexists=Test-Path $logdir
   if (!($logdirexists)) {
       mkdir $logdir 2>&1 | out-null
    }

   $logfileexists=Test-Path $logfile
   if ($logfileexists) {
       rm $logfile 2>&1 | out-null
    }

   $filelistexists=Test-Path $filelist
   if ($filelistexists) {
       rm $filelist 2>&1 | out-null
    }
}

function ProcessCommand
{
```

```
   $op=$args
   echo "EPM Automate operation: epmautomate.bat $op" >> $logfile
   if ($op -eq 'listfiles') {
      epmautomate.bat $op | where {$_ -like ' apr/*/access_log.zip'} | Tee-
Object -FilePath $filelist | Out-File $logfile -Append 2>&1
    } else {
      epmautomate.bat $op >> $logfile 2>&1
        if ($LASTEXITCODE -ne 0) {
           echo "EPM Automate operation failed: epmautomate.bat $op.
See $logfile for details."
           #exit
           }
    }
}

function RunEpmAutomateCommands
{
    EchoAndLogMessage "Running EPM Automate commands to anonymize data in
the access logs and activity reports."
    ProcessCommand login $username $password $url
    ProcessCommand listfiles
    ProcessFiles
    ProcessCommand logout
}

function ProcessActivityReport
{
    $activityreport=$args[0]
    $user=$args[1]

    $activityreportexists=Test-Path "$activityreport"
    if ($activityreportexists) {
        LogMessage "Removing User ID: $user from activity
report $activityreport"
        (Get-Content "$activityreport").replace("$user", 'XXXXX') | Set-
Content "$activityreport"
        $txt = [io.file]::ReadAllText("$activityreport") -replace
"`r`n","`n"
        [io.file]::WriteAllText("$activityreport", $txt)
        #Get-ChildItem -File -Recurse | % { $x = get-content -raw -
path $activityreport; $x -replace "`r`n","`n" | set-content -
path $activityreport }
    }
}

function AnonymizeData
{
    $aprdir=$args[0]
    $datestampdir=$args[1]
    $path="$aprdir/$datestampdir"
    $accesslogzipped="access_log.zip"
    $accesslog="access_log.csv"
    $accesslogupdated=$accesslog + ".tmp"
    $activityreportfile="$datestampdir" + ".html"
    $userArray = @()
```

```
    expand-Archive -Path "$path/$accesslogzipped" -DestinationPath $path
    rm $path/$accesslogzipped 2>&1 | out-null
    $accesslogexists=Test-Path "$path/$accesslog"
    if ($accesslogexists) {
        EchoAndLogMessage "Processing access log: $path/$accesslog"
        Get-Content $path/$accesslog | ForEach-Object {
            $elements=[regex]::Split( $_ , ',(?=(?:[^"]|"[^"]*")*$)' )
            $date=$elements[0]
            $time=$elements[1]
            $uri=$elements[2]
            $duration=$elements[3]
            $bytes=$elements[4]
            $ip=$elements[5]
            $user=$elements[6]
            $screen=$elements[7]
            $action=$elements[8]
            $object=$elements[9]
            if ($date -like 'Date') {
                echo "$_" >> $path/$accesslogupdated
            } else {
                if ($user -notlike '-') {
                    LogMessage "Removing instance of User ID: $user
from $path/$accesslog."
                    echo
"$date,$time,$uri,$duration,$bytes,$ip,XXXXX,$screen,$action,$object"
>> $path/$accesslogupdated
                    $userArray += $user
                } else {
                    echo
"$date,$time,$uri,$duration,$bytes,$ip,$user,$screen,$action,$object"
>> $path/$accesslogupdated
                }
            }
        }
        #Get-ChildItem -File -Recurse | % { $x = get-content -raw -
path $path/$accesslogupdated; $x -replace "`r`n","`n" | set-content -
path $path/$accesslogupdated }
        $txt = [io.file]::ReadAllText("$path/$accesslogupdated") -replace
"`r`n","`n"
        [io.file]::WriteAllText("$path/$accesslogupdated", $txt)
        mv -Force $path/$accesslogupdated $path/$accesslog
        Compress-Archive -Path $path/$accesslog $path/$accesslogzipped
        rm $path/$accesslog 2>&1 | out-null
    }

    EchoAndLogMessage "Processing activity
report: $path/$activityreportfile"
    $userArray = $userArray | Select-Object -Unique
    foreach ($element in $userArray) {
        ProcessActivityReport "$path/$activityreportfile"
"$element"
    }
}

function ProcessFiles
{
```

```
    # Loop through iteration csv file and parse
    Get-Content $filelist | ForEach-Object {
        $fullpath=$_.trim()
        $elements=$fullpath.split('/')
        $aprdir=$elements[0]
        $datestampdir=$elements[1]
        $accesslogfile="access_log.zip"
        $activityreportfile="$datestampdir" + ".html"
        $datestampdirexists=Test-Path "$aprdir/$datestampdir"
        $accesslog="$aprdir/$datestampdir/$accesslogfile"
        $activityreport="$aprdir/$datestampdir/$activityreportfile"

        echo "fullpath: $fullpath" >> $logfile
        echo "aprdir: $aprdir, datestampdir: $datestampdir" >> $logfile
        if (!($datestampdirexists)) {
            mkdir "$aprdir/$datestampdir" -ea 0 2>&1 | out-null
            ProcessCommand downloadfile "$accesslog"
            ProcessCommand downloadfile "$activityreport"
            mv "$accesslogfile" "$aprdir/$datestampdir"
            mv "$activityreportfile" "$aprdir/$datestampdir"
            AnonymizeData "$aprdir" "$datestampdir"
            ProcessCommand deletefile "$accesslog"
            ProcessCommand deletefile "$activityreport"
            ProcessCommand uploadfile "$accesslog" "$aprdir/$datestampdir"
            ProcessCommand uploadfile "$activityreport"
"$aprdir/$datestampdir"
        } else {
            EchoAndLogMessage "Files in directory $aprdir/$datestampdir
were processed earlier. Skipping these files."
        }
    }
}

function callSendMail
{
    $elements=$logfile.split('/')
    $logfilename=$elements[3]

    if (${emailtoaddress} -match "@") {
        epmautomate.bat login ${username} ${password} ${url}
        epmautomate.bat uploadFile "$logfile"
        epmautomate.bat sendMail $emailtoaddress "Mask Access Logs and
Activity Reports results" Body="The results of running the Mask Access
Logs and Activity Reports script are attached." Attachments=$logfilename
        epmautomate.bat deleteFile "$logfilename"
        epmautomate.bat logout
    }
}

Init
EchoAndLogMessage "Starting the anonymize data script"
RunEpmAutomateCommands
EchoAndLogMessage "Anonymize data script completed"
EchoAndLogMessage "Refer to logfile: $logfile for details."
callSendMail
```

3. Using Windows Scheduler, schedule `anonymizeData.bat`. See Automating Script Execution for detailed steps.

   You need to supply the following parameter values to execute `anonymizeData.bat`

   - User name of a Service Administrator
   - Password of the Service Administrator or the location where the encrypted password file is available
   - URL of the service environment in which the Access Logs and Activity Reports are to be masked
   - **Optional:** The email address to which the report is to be sent. The report is emailed only if this value is specified.

Automate Activity Report Downloads to a Local Computer

Use the script in this section to automate the downloading of Activity Reports from an environment to a local computer.

You use `syncAprReports.bat` to download Activity Reports. You can schedule the batch file using Windows scheduler to automate the downloading of Activity Reports. See Using Activity Reports and Access Logs to Monitor Usage in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators* for detailed information about Activity Report.

You manually create `syncAprReports.bat` by copying the script provided in the following procedure and then updating the connection parameters. This script checks the environment and downloads only the reports that are more recent than those available in the download directory on the local computer.

> **Note:**
>
> - The script is to be run from a Windows computer only.
> - This script does not download the Feedback Activity Report, which is generated when users submit feedback.
> - If the password you use contains special characters, see Handling Special Characters

1. Create a batch (`.BAT`) file named `syncAprReports.bat` containing the following script and save it in a convenient location, for example, `C:\automate_scripts`.

```
@echo off
title APR
setlocal DisableDelayedExpansion

REM To hardcode the values in the script replace %1, %2, %3, and %4, with
the actual values.
REM Example:
REM set apr_dir="C:\Oracle\apr"
REM set username="serviceAdmin"
REM set password="Ex@mple!"
REM set url="https://test-example.stg-pbcs.us1.oraclecloud.com"
```

```
set apr_dir=%1
set username=%2
set password=%3
set url=%4
setlocal EnableDelayedExpansion
set epmautomate_dir=%cd%
set lastfile=
set argC=0
for %%x in (%*) do Set /A argC+=1
if %argC% neq 0 (
    if %argC% neq 3 (
        if %argC% neq 4 (
            goto :usage
        )
    )
)
goto :login
:usage
echo.
echo Invalid syntax. Please check the parameters.
echo Syntax:
echo 1) syncAprReports.bat APR_FolderPath_on_client username password url
echo      or
echo 2) set the parameters in the file and use below syntax
echo    syncAprReports.bat
goto :end

:login
setlocal DisableDelayedExpansion
for /f "delims=" %%i in ('epmautomate login %username% %password% %url%')
do set result=%%i
if "Login successful" neq "%result%" (
    echo Login Failed
    goto :end
)

if not exist %apr_dir% (
echo.
echo apr folder does not exist
GOTO :end
)
cd /D %apr_dir%
for /f "delims=" %%D in ('dir /a:d /b /o:-n') do (
REM AFTER: for /f "delims=" %%D in ('dir /a-d /b /s /o:-n') do (
    set "lastFile=%%~nD"
    goto :next
)

:next
setlocal EnableDelayedExpansion
echo.
echo Most Recent APR on client is %lastFile%

set "output_cnt=0"
cd /D %epmautomate_dir%
for /F "delims=" %%f in ('epmautomate listfiles') do (
```

```
    cd /D !apr_dir!
    set "line=%%f"
    for /f "tokens=* delims= " %%a in ("!line!") do set line=%%a
    if "!line:~0,3!" equ "apr" (

        if "!line:~4,8!" neq "Feedback" (

            set isValidFile=false
            if "!line:~-5!" equ ".html" set isValidFile=true
            if "!line:~-5!" equ ".json" set isValidFile=true

            if "!isValidFile!" equ "true" (

                if "%lastFile%" lss "!line:~4,19!" (

                        if "!line:~4,19!" neq "!dirname!" (

                            set apr_dir=!apr_dir:"=!
                            set /a output_cnt+=1
                            set "output[!output_cnt!]=!apr_dir!\!
line:~4,19!"
                            set "dirname=!line:~4,19!"

                            REM start downloading
                            mkdir "!dirname!"
                            cd /D !dirname!
                            echo downloading !line!
                            set "downloadDir=!apr_dir!\!dirname!"

                            cd /D %epmautomate_dir%
                            for /f "delims=" %%i in ('epmautomate
downloadfile "!line!"') do set result1=%%i
                            move "!line:~24!" "!downloadDir!" > nul
                            echo !result1!
                            REM end downloading

                        ) else (

                            REM start downloading
                            cd /D !dirname!
                            echo downloading !line!
                            set apr_dir=!apr_dir:"=!
                            set "downloadDir=!apr_dir!\!dirname!"
                            cd /D %epmautomate_dir%
                            for /f "delims=" %%i in ('epmautomate
downloadfile "!line!"') do set result1=%%i
                            move "!line:~24!" "!downloadDir!" > nul
                            echo !result1!
                            REM end downloading

                        )
                ) else (

                    REM TO-DO
```

```
                    )
                )
            )
        )
)

echo.
echo %output_cnt% APR's downloaded
for /L %%n in (1 1 !output_cnt!) DO echo !output[%%n]!
goto :end

:end
cd /D %epmautomate_dir%
endlocal
```

2. Modify `syncAprReports.bat` to set the values for the parameters in the following table. These values are used to access the environment to download Activity Reports.

Table 3-3 Parameter Values to Include in `syncAprReports.bat`

Parameter	Expected Value
`set apr_dir=%1`	Specify an existing directory into which Activity Reports are to be downloaded. **Example:** `set apr_dir="C:\Oracle\apr"`
`set username=%2`	An Oracle Enterprise Performance Management Cloud user name that is to be used to sign into the environment to download Activity Reports. **Example:** `set username="ServiceAdmin"`

`set password=%3`	The name and location of the file that stores the encrypted password of the user specified by the `username` variable. You may also specify the plain text password of the user (not recommended). See the encrypt command for information on creating an encrypted password file.
	Examples:
	`set password="C:\mySecuredir\password.epw"`
	`set password="Ex@mple1"`

`set url=%4`	The URL of the environment. **Example:** `set url="https://test-example.stg-pbcs.us1.oraclecloud.com"`

3. Using Windows Scheduler, schedule `syncAprReports.bat`. See Automating Script Execution for detailed steps.

Download Access Logs from an Environment

Use the script in this section to automate the process of downloading access logs from an environment to a local computer.

You can schedule `syncAccessLog.bat` using Windows scheduler to automate the downloading of the log files. See Viewing and Downloading Activity Reports and Access Logs in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators* for procedures to download access logs using Application Management.

The following script checks the environment and downloads only the log files that are more recent than those available in the download directory on the local computer. The is a Windows script; you can create a similar shell script for Linux/UNIX environments.

1. Create a batch (.BAT) file named `syncAccessLog.bat` containing the following script and save it in a convenient location, for example, `C:\automate_scripts`.

> **Note:**
>
> If your password contains special characters, see Handling Special Characters.

```
@echo off
title APR
setlocal DisableDelayedExpansion

REM To hardcode the values in the script replace %1, %2, %3, and %4 with
the actual values.
REM Example:
REM set apr_dir="C:\Oracle\apr"
REM set username="serviceAdmin"
REM set password="C:\mySecuredir\password.epw"
REM set url="https://test-cloudpln.pbcs.us1.oraclecloud.com"
set apr_dir=%1
set username=%2
set password=%3
set url=%4

setlocal EnableDelayedExpansion
set epmautomate_dir=%cd%
set lastfile=
REM if [%1]==[] goto :usage
REM if [%2]==[] goto :usage
REM if [%3]==[] goto :usage

set argC=0
for %%x in (%*) do Set /A argC+=1
if %argC% neq 0 (
  if %argC% neq 3 (
    if %argC% neq 4 (
      goto :usage
      )
  )
)
goto :login

:usage
echo.
echo Invalid syntax. Please check the parameters.
echo Syntax:
echo 1) syncAccessLog.bat APR_FolderPath_on_client username password url
echo     or
echo 2) set the parameters in the file and use below syntax
echo syncAccessLog.bat
goto :end
```

```
:login
setlocal DisableDelayedExpansion
REM for /f "delims=" %%i in ('epmautomate login %2 %3 %4') do set result=%
%i
for /f "delims=" %%i in ('epmautomate login %username% %password% %url%')
do set result=%%i

if not exist %apr_dir% (
echo.
echo apr folder does not exist
GOTO :end
)
cd /D %apr_dir%
for /f "delims=" %%D in ('dir /a:d /b /o:-n') do (
REM AFTER: for /f "delims=" %%D in ('dir /a-d /b /s /o:-n') do (
  set "lastFile=%%~nD"
  goto :next
)

:next
setlocal EnableDelayedExpansion
echo.
echo Most Recent Access Log on client is %lastFile%

set "output_cnt=0"
cd /D %epmautomate_dir%
for /F "delims=" %%f in ('epmautomate listfiles') do (

  cd /D !apr_dir!
  set "line=%%f"
  for /f "tokens=* delims= " %%a in ("!line!") do set line=%%a
  if "!line:~0,3!" equ "apr" (
    if "!line:~-4!" equ ".zip" (
      if "%lastFile%" lss "!line:~4,19!" (
          if "!line:~4,19!" neq "!dirname!" (
            set apr_dir=!apr_dir:"=!
            set /a output_cnt+=1
            set "output[!output_cnt!]=!apr_dir!\!line:~4,19!"
            set "dirname=!line:~4,19!"

            REM start downloading
            mkdir "!dirname!"
            cd /D !dirname!
            echo downloading !line!
            set "downloadDir=!apr_dir!\!dirname!"
            cd /D %epmautomate_dir%
            for /f "delims=" %%i in ('epmautomate downloadfile "!line!"')
do set result1=%%i
            move "!line:~24!" "!downloadDir!" > nul
            echo !result1!
            REM end downloading

          ) else (
            REM start downloading
            cd /D !dirname!
```

```
            echo downloading !line!
            set apr_dir=!apr_dir:"=!
            set "downloadDir=!apr_dir!\!dirname!"
            cd /D %epmautomate_dir%
            for /f "delims=" %%i in ('epmautomate downloadfile "!line!"')
do set result1=%%i
            move "!line:~24!" "!downloadDir!" > nul
            echo !result1!
            REM end downloading
          )
      ) else (
        REM TO-DO
      )
    )
  )
)

echo.
echo %output_cnt% access logs downloaded
for /L %%n in (1 1 !output_cnt!) DO echo !output[%%n]!
goto :end

:end
cd /D %epmautomate_dir%
endlocal
```

2. Modify `syncAccessLog.bat` to set the values for the parameters in the following table. These values are used to access the environment to download access logs.

Table 3-4 Variable Values to Include in `syncAccessLog.bat`

Variable	Expected Value
`set apr_dir=%1`	Specify the directory into which access logs are to be downloaded. **Example:** `set apr_dir="C:\Oracle\apr"`
`set username=%2`	An Oracle Enterprise Performance Management Cloud user name that is to be used to sign into the environment to download access logs. **Example:** `set username="ServiceAdmin"`

`set password=%3`	The name and location of the file that stores the encrypted password of the user specified by the `username` variable. You may also specify the plain text password of the user (not recommended). See the encrypt command for information on creating an encrypted password file.
	Examples:
	`set password="C:\mySecuredir\password.epw"`
	`set password="P@ssword1"`

`set url=%4`	The URL of the environment. **Example:** `set url="https://test-cloudpln.pbcs.us1.oraclecloud.com"`

3. Using Windows Scheduler, schedule `syncAccessLog.bat`. See Automating Script Execution for detailed steps.

Automate the Cloning of Environments

Use the script in this section to automate the cloning of environments.

Create a batch (`.bat`) or shell (`.sh`) file containing script similar to the following to clone an environment. The following sample scripts handle these activities:

- Sign in to the source environment.
- Use `Artifact Snapshot` (the snapshot created during the last daily maintenance of the source environment) or another snapshot available in the source environment to convert the target environment as a clone of the source.
- Optionally, create users and their predefined and application role assignments matching those in the source environment.
- Optionally, change the daily maintenance start time to match that of the source environment.
- Sign out.

For detailed information on cloning process, see "Cloning EPM Cloud Environments " in *Administering Migration for Oracle Enterprise Performance Management Cloud*.

See Automating Script Execution for information on scheduling the script using Windows Task Scheduler.

Windows

1. Create a batch (.BAT) file named `cloneEnvironment.bat` containing the following script and save it in a convenient location, for example, `C:\automate_scripts`.

```
@echo off
set paramRequiredMessage=Syntax: cloneEnvironment.bat "SOURCE USERNAME"
"SOURCE PASSWORD FILE" "SOURCE URL" "TARGET USERNAME" "TARGET PASSWORD
FILE" "TARGET URL"

set usersandpredefinedroles="false"
set snapshotname="Artifact Snapshot"
set dailymaintenancestarttime="true"
set dirpath=%~dp0
cd %dirpath:~0,-1%

if "%~1" == "" (
  echo Source User Name is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~2" == "" (
  echo Source Password File is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~3" == "" (
  echo Source URL is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
```

```
if "%~4" == "" (
  echo Target User Name is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~5" == "" (
  echo Target Password File is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~6" == "" (
  echo Target URL is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )

PowerShell.exe -File cloneEnvironment.ps1 %~1 %~2 %~3 %~4 %~5 %~6
%usersandpredefinedroles% %snapshotname% %dailymaintenancestarttime%
```

2. Modify `cloneEnvironment.bat` to set the values for these parameters:

Table 3-5 Parameters to set in cloneEnvironment.bat

Parameter	Description
`usersandpredefinedroles`	Set the value of this parameter to `true` to clone users and their predefined and application role assignments. To clone users and role assignments, the user running the script must have the Service Administrator role and the Identity Domain Administrator in the target environment.
`snapshotname`	Name of the snapshot in the source environment that should be used for cloning.
`dailymaintenancestarttime`	Set the value of this parameter to `true` to change the daily maintenance start time of the cloned environment to that of the source environment. Set this value to `false` to retain the current daily maintenance start time of the cloned environment.

3. Create a PowerShell script named `cloneEnvironment.ps1` containing the following script and save it in the directory where you saved `cloneEnvironment.bat`, for example, `C:\automate_scripts`.

```
# Clone Environment script

$source_username=$args[0]
$source_password=$args[1]
$source_url=$args[2]
$target_username=$args[3]
$target_password=$args[4]
$target_url=$args[5]
$usersandpredefinedroles=$args[6]
$snapshotname=$args[7]
$dailymaintenancestarttime=$args[8]

epmautomate.bat login "${source_username}" "${source_password}" "$
```

```
{source_url}"
epmautomate.bat cloneEnvironment "${target_username}" "${target_password}"
"${target_url}" UsersAndPreDefinedRoles="${usersandpredefinedroles}"
SnapshotName="${snapshotname}" DailyMaintenanceStartTime="$
{dailymaintenancestarttime}"
epmautomate.bat logout
```

4. Run `cloneEnvironment.bat` using this command:

   ```
   cloneEnvironment.bat "SOURCE USERNAME" "SOURCE PASSWORD FILE" "SOURCE URL"
   "TARGET USERNAME" "TARGET PASSWORD FILE" "TARGET URL"
   ```

   For example:

   ```
   cloneEnvironment.bat jdoe@example.com C:\mySecuredir\example_pwd.epw
   https://source_example.oraclecloud.com jdoe@example.com
   C:\mySecuredir\example_pwd2.epw https://target_example.oraclecloud.com.
   ```

Linux

1. Create a shell script named `cloneEnvironment.sh` containing the following script and save it in a convenient location.

   ```
   #!/bin/bash

   # Update the following parameters
   # -------------------------------
   epmautomatescript=/home/user1/epmautomate/bin/epmautomate.sh
   javahome=/home/user1/jdk1.8.0_191/
   usersandpredefinedroles="false"
   snapshotname="Artifact Snapshot"
   dailymaintenancestarttime="true"
   # -------------------------------

   source_username="$1"
   source_password="$2"
   source_url="$3"
   target_username="$4"
   target_password="$5"
   target_url="$6"

   export JAVA_HOME=${javahome}

   if [ "$#" -ne 6 ]; then
       echo "Usage: ./cloneEnvironment.sh <SOURCE USERNAME> <SOURCE PASSWORD
   FILE> <SOURCE URL> <TARGET USERNAME> <TARGET PASSWORD FILE> <TARGET URL>"
       exit 1
   fi

   ${epmautomatescript} login "${source_username}" "${source_password}" "$
   {source_url}"
   ${epmautomatescript} cloneEnvironment "${target_username}" "$
   {target_password}" "${target_url}" UsersAndPreDefinedRoles="$
   {usersandpredefinedroles}" SnapshotName="${snapshotname}"
   ```

```
DailyMaintenanceStartTime="${dailymaintenancestarttime}"
${epmautomatescript} logout
```

2. Modify `cloneEnvironment.sh` to set the values for these parameters:

Table 3-6 Parameters to set in cloneEnvironment.sh

Parameter	Description
`epmautomatescript`	The absolute path of EPM Automate executable (`epmautomate.sh`).
`javahome`	JAVA_HOME location.
`usersandpredefinedroles`	Set the value of this parameter to `true` to clone users and their predefined and application role assignments. To clone users and role assignments, the user running the script must have the Service Administrator role and the Identity Domain Administrator in the target environment.
`snapshotname`	Name of the snapshot in the source environment that should be used for cloning.
`dailymaintenancestarttime`	Set the value of this parameter to `true` to change the daily maintenance start time of the cloned environment to that of the source environment. Set this value to `false` to retain the current daily maintenance start time of the cloned environment.

3. Run `cloneEnvironment.sh`.

```
./cloneEnvironment.sh "SOURCE USERNAME" "SOURCE PASSWORD FILE" "SOURCE
URL" "TARGET USERNAME" "TARGET PASSWORD FILE" "TARGET URL"
```

For example:

```
./cloneEnvironment.sh jdoe@example.com ./home/secure/example_pwd.epw
https://source_example.oraclecloud.com jdoe@example.com ./home/secure/
example_pwd.epw2 https://target_example.oraclecloud.com.
```

Clone from Primary to Standby Environment Daily After Daily Maintenance is Complete on the Primary Environment

To keep the standby environment up to date with the primary environment, use these scripts to clone Oracle Enterprise Performance Management Cloud primary environment to the standby environment soon after the daily maintenance is complete on the primary environment.

These custom scripts identify whether the scheduled daily maintenance for the day is complete and then clone the environment.

Windows Script

Create `dailyclone.ps1` by copying the following PowerShell script.

```
# Clone Environment script
#
# Update the following parameters
```

```
# ------------------------------
$users_and_predefined_roles="false"
$daily_maintenance_start_time="true"
$data_management="true"
$app_audit="true"
$job_console="true"
$stored_snapshots_and_files="false"
# ------------------------------

$source_username=$args[0]
$source_password=$args[1]
$source_url=$args[2]
$target_username=$args[3]
$target_password=$args[4]
$target_url=$args[5]

if ($($args.count) -ne 6) {
    echo "Usage: ./dailyclone.ps1 <SOURCE USERNAME> <SOURCE PASSWORD> <SOURCE
URL> <TARGET USERNAME> <TARGET PASSWORD> <TARGET URL>"
    exit 1
}

$amw_time=""

function getDailyMaintenanceStartTime {
    $amwstring=$(epmautomate.bat getDailyMaintenanceStartTime)
    $elements=$amwstring.split(' ')
    $amwtime=$elements[0]
    return $amwtime
}

function goToSleep ($amw_time){
    $current_mdy=Get-Date -AsUTC -UFormat "%m/%d/%Y"
    $current_date_time=Get-Date -AsUTC -UFormat "%m/%d/%Y %H:%M:%S"
    $current_epoch=Get-Date -Date $current_date_time -UFormat "%s"
    $target_date_time=[DateTime]"${current_mdy} ${amw_time}"
    $target_epoch=Get-Date -Date $target_date_time -UFormat "%s"
    $sleep_seconds=$target_epoch - $current_epoch

    # Today's AMW start time has already passed, so add 24 hours to
sleep_seconds
    if ($sleep_seconds -lt 0) {
        $sleep_seconds=$sleep_seconds + 86400
    }

    $sleep_ts=New-TimeSpan -Seconds ${sleep_seconds}
    $sleep_hms="${sleep_ts}" -replace '^\d+?\.'

    echo "Current time is ${current_date_time}. Sleeping for ${sleep_hms},
until daily maintenance start time of ${amw_time}."
    Start-Sleep -Seconds $sleep_seconds
}

function deleteArtifactSnapshotIfExists {
    if (artifactSnapshotExists) {
        $command_del=$(epmautomate.bat deletefile "Artifact Snapshot")
```

```
    }
}

function artifactSnapshotExists {
    $filelist=$(epmautomate.bat listfiles)
    if ("$filelist".contains("Artifact Snapshot")) {
        return $true
    else
        return $false
    }
}

function cloneEnvironment {
    echo "Checking to see if daily maintenance processing has completed ..."
    while ($true) {
        if (artifactSnapshotExists) {
            echo "Daily maintenance processing has completed ..."
            break
    } else {
            echo "Sleeping for 30 seconds before the next check to see if
daily maintenance processing has completed ..."
            Start-Sleep -Seconds 30
        }
    }

    echo "Encrypting target password ..."
    epmautomate.bat encrypt "${target_password}" "oracleKey"
"target_password.epw"

    echo "Cloning environment ..."
    epmautomate.bat cloneEnvironment "${target_username}"
"target_password.epw" "${target_url}" "SnapshotName=Artifact Snapshot"
"UsersAndPreDefinedRoles=${users_and_predefined_roles}" "DataManagement=$
{data_management}" "appAudit=${app_audit}" "jobConsole=${job_console}"
"storedSnapshotsAndFiles=${stored_snapshots_and_files}"
"dailyMaintenanceStartTime=${daily_maintenance_start_time}"
}

echo "Beginning clone environment script."
echo "Logging into server ..."
epmautomate.bat login ${source_username} ${source_password} ${source_url}
$amwtime=getDailyMaintenanceStartTime
goToSleep ($amwtime)
deleteArtifactSnapshotIfExists
cloneEnvironment
echo "Logging out of server ..."
epmautomate.bat logout
echo "Clone environment script processing has completed."
```

Linux/UNIX Script

Create `dailyclone.sh` by copying the following script.

```
#!/bin/bash

# Update the following parameters
```

```
# ------------------------------
epmautomatescript="LOCATION_EPM_AUTOMATE_EXECUTABLE"
javahome="LOCATION_JAVA_HOME"
users_and_predefined_roles="false"
data_management="true"
app_audit="true"
job_console="true"
stored_snapshots_and_files="false"
daily_maintenance_start_time="true"
# ------------------------------

source_username="$1"
source_password="$2"
source_url="$3"
target_username="$4"
target_password="$5"
target_url="$6"

export JAVA_HOME=${javahome}

if [ "$#" -ne 6 ]; then
    echo "Usage: ./dailyclone.sh SOURCE_USERNAME SOURCE_PASSWORD SOURCE_URL
TARGET_USERNAME TARGET_PASSWORD TARGET_URL"
    exit 1
fi

amw_time=""

getDailyMaintenanceStartTime() {
    amw_time=$(${epmautomatescript} getDailyMaintenanceStartTime | cut -d' ' -
f1)
}

goToSleep() {
    current_mdy=$(date -u +%m/%d/%Y)
    current_date_time=$(date -u)
    current_epoch=$(date +%s)
    target_epoch=$(date -d "${current_mdy} ${amw_time}" +%s)
    sleep_seconds=$(($target_epoch - $current_epoch))

    # Today's AMW start time has already passed, so add 24 hours to
sleep_seconds
    if [[ ${sleep_seconds} -lt 0 ]]
    then
        sleep_seconds=$((sleep_seconds + 86400))
    fi

    sleep_hms=$(date -d@${sleep_seconds} -u +%H:%M:%S)

    echo "Current time is ${current_date_time}. Sleeping for ${sleep_hms},
until daily maintenance start time of ${amw_time}."
    sleep $sleep_seconds
}

deleteArtifactSnapshotIfExists() {
    found=1
```

```
    filelist=$(${epmautomatescript} listfiles)
    if [[ ${filelist} == *"Artifact Snapshot"* ]]
    then
        command_del=$(${epmautomatescript} deletefile "Artifact Snapshot")
    fi
}

artifactSnapshotExists() {
    found=1

    filelist=$(${epmautomatescript} listfiles)
    if [[ ${filelist} == *"Artifact Snapshot"* ]]
    then
        found=0
    else
        found=1
    fi

    echo ${found}
}

cloneEnvironment() {
    local found=1

    while true
    do
        found=$(artifactSnapshotExists)
        if [[ ${found} -eq 0 ]]
        then
            echo "Daily maintenance processing has completed ..."
            break
        else
            echo "Sleeping for 30 seconds before the next check to see if
daily maintenance processing has completed ..."
            sleep 30
        fi
    done

    echo "Encrypting target password ..."
    ${epmautomatescript} encrypt "${target_password}" "oracleKey"
"target_password.epw"

    echo "Cloning environment ..."
    ${epmautomatescript} cloneEnvironment "${target_username}"
"target_password.epw" "${target_url}" "SnapshotName=Artifact Snapshot"
"UsersAndPreDefinedRoles=${users_and_predefined_roles}" "DataManagement=$
{data_management}" "appAudit=${app_audit}" "jobConsole=${job_console}"
"storedSnapshotsAndFiles=${stored_snapshots_and_files}"
"dailyMaintenanceStartTime=${daily_maintenance_start_time}"
}

echo "Beginning clone environment script."
echo "Logging into server ..."
${epmautomatescript} login ${source_username} ${source_password} ${source_url}
getDailyMaintenanceStartTime
goToSleep
```

```
deleteArtifactSnapshotIfExists
cloneEnvironment
echo "Logging out of server ..."
${epmautomatescript} logout
echo "Clone environment script processing has completed."
```

Running the Script

- Create `dailyclone.ps1` or `dailyclone.sh`by copying a script from one of the preceding sections.
- Update these values in `dailyclone.sh`:
  - `epmautomatescript` with the location of the EPM Automate executable. Example: `epmautomatescript="/home/utils/EPMAutomate/bin/epmautomate.sh"`
  - `javahome` with the directory where the JDK used by EPM Automate is installed. For example: `"/home/user1/jdk1.8.0_191"`
- Update these values in `dailyclone.ps1` and `dailyclone.sh`, if required:
  - `users_and_predefined_roles`: Set this value to `true` to clone users and their predefined role assignments (Access Control groups are always cloned).
  - `data_management`: Set this value to `false` to not clone Data Integration records. Note that Data Integration records can be cloned only if both the source and target environments are on the same monthly update or the target environment is one update newer than the source environment. For example, you can clone 22.01 Data Management records to another 22.01 environment or to a 22.02 environment only. Ignored for Narrative Reporting and Oracle Enterprise Data Management Cloud environments.
  - `app_audit`: Set this value to `false` if you do not want to clone the application audit data for Planning, FreeForm, and Enterprise Profitability and Cost Management applications.
    Financial Consolidation and Close and Tax Reporting audit information is always cloned.
  - `job_console`: Set this value to `false` if you do not want to clone job console data.
  - `stored_snapshots_and_files`: Set this value to `true` if you want to clone the contents of the top level folders in the inbox and outbox (subfolders are never cloned) of the source environment.
  - `daily_maintenance_start_time`: Set this value to `false` if you do not want to reset the maintenance start time of the cloned target environment to that of the source environment.
- Run `dailyclone.ps1` or `dailyclone.sh`: In a Command Window, or shell, navigate to the folder where `dailyclone.ps1` or `dailyclone.sh` is stored and then execute a command:
  - **Windows**: `./dailyclone.ps1` *`SOURCE_USERNAME SOURCE_PASSWORD SOURCE_URL TARGET_USERNAME TARGET_PASSWORD TARGET_URL`*
  - **Linux/UNIX**: `./dailyclone.sh` *`SOURCE_USERNAME SOURCE_PASSWORD SOURCE_URL TARGET_USERNAME TARGET_PASSWORD TARGET_URL`* where:
    - `SOURCE_USERNAME` is the user name of a Service Administrator. Identity Domain Administrator role is required to clone users and predefined roles.
    - `SOURCE_PASSWORD` is the password of the user identified by *`SOURCE_USERNAME`*.
    - `SOURCE_URL` is the URL of the environment that you want to clone.

* `TARGET_USERNAME` is the user name of a Service Administrator. Identity Domain Administrator role is required to clone users and predefined roles.
* `TARGET_PASSWORD` is the password of the user identified by *`TARGET_USERNAME`*.
* `TARGET_URL` is the URL of the target environment.

Remove Unnecessary Files from an Environment

Use these scripts to remove unnecessary files from an environment.

These scripts perform the following steps:

- Signs in to the environment.
- Lists the files and snapshots in the environment.
- Deletes the files specified in `input.properties`.
- Signs out.

Windows Sample Script

Create a file named `removeUnnecessaryFiles.ps1` by copying the following script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$file1="$($inputproperties.file1)"
$file2="$($inputproperties.file2)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate listfiles
epmautomate deletefile ${file1}
epmautomate deletefile ${file2}
epmautomate logout
```

Linux/UNIX Sample Script

Create a file named `removeUnnecessaryFiles.sh` by copying the following script. Store it in a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} listfiles
${epmautomatescript} deletefile "${file1}"
${epmautomatescript} deletefile "${file2}"
${epmautomatescript} logout
```

Creating the input.properties File

To run the `removeUnnecessaryFiles` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `removeUnnecessaryFiles.ps1` or `removeUnnecessaryFiles.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
file1=FILE_NAME
file2=FILE_NAME
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
file1=FILE_NAME
file2=FILE_NAME
```

Table 3-7 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator, who also has the Identity Domain Administrator role.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment from which you want to generate the snapshot.
`file1` and `file2`	Name of a file or snapshot to delete from the environment. If the file is not in the Outbox, specify the path and name of the file.

Running the Script

1. Create `removeUnnecessaryFiles.ps1` or `removeUnnecessaryFiles.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `removeUnnecessaryFiles` script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run `removeUnnecessaryFiles.ps1`.
   - **Linux/UNIX:** run `./removeUnnecessaryFiles.sh`.

Find and Download Files from an Environment

Use the sample script in this section to automate the process of downloading one or more files from an Oracle Enterprise Performance Management Cloud environment using a text string as a wildcard.

The following script allows you to match the string that you specify as the value of the `FILENAME` parameter with file names displayed using the `listfiles` command and then automatically download the files that match the string.

Be sure to assign the appropriate search string to the `FILENAME` parameter. For example, `FILENAME="Scheduler Output/epm"` will match the string `Scheduler Output/epm` against file names in the `listfiles` command output in your environment to identify the files to download.

Input parameters for running this script are `username`, `password` or `password_file`, and `service_url`.

> **Note:**
>
> If the password contains special characters, see Handling Special Characters.

Windows

```
@echo off
    setlocal EnableExtensions EnableDelayedExpansion
    set USERNAME="username"
    set PASSWORD="password"
    set URL="url"

call epmautomate login %USERNAME% %PASSWORD% %URL%
    set FILENAME="Scheduler Output/epm"
    for /f "tokens=*" %%i in ('epmautomate listfiles ^| findstr /b /r /c:"^
*%FILENAME%" ') do (
    call epmautomate downloadfile "%%i"
    )
call epmautomate logout
endlocal
```

Linux/UNIX

```
#!/bin/sh
    USERNAME="username"
    PASSWORD="password"
    URL="url"

./epmautomate.sh login $USERNAME $PASSWORD $URL
    FILENAME='Scheduler Output/epm'
    #echo $FILENAME
./epmautomate.sh listfiles | grep "^ $FILENAME" | while read -r line ; do
    echo "Processing $line"
```

```
    ./epmautomate.sh downloadfile "$line"
    done
./epmautomate.sh logout
```

Recreate an Old EPM Cloud Environment for Audits

Use the script in this section to create a self-service solution to maintain an up-to-date library of snapshots for your Oracle Enterprise Performance Management Cloud environment. You require an environment dedicated for the purpose of upgrading and maintaining a library of up-to-date snapshots.

EPM Cloud supports snapshot compatibility for one monthly cycle only; you can migrate maintenance snapshots from the test environment to the production environment and vice versa. However, the auditing requirements of some customers may necessitate restoring snapshots from multiple years on the latest environment, and accessing application in a short period of time.

You should schedule this script to run once every month to convert the available snapshots and make them compatible with the latest EPM Cloud patch level. Oracle recommends that you run the script after the third Friday of the month to ensure that all issues within the production environment have been resolved.

> **Note:**
>
> You cannot use this script to update Narrative Reporting, Account Reconciliation, and Oracle Enterprise Data Management Cloud snapshots.

How the Script Works

For every snapshot stored by the customer, the upgrade script completes these tasks using EPM Automate :

1. Using the information in the `input.properties` file, logs into an environment
2. Uses the `recreate` command to refurbish the environment
3. Imports the snapshot into the environment
4. Runs daily maintenance on the environment, which results in the snapshot being converted into the format compatible with the current EPM Cloud patch level.
5. Downloads `Artifact Snapshot` (the maintenance snapshot) into a folder. If you re-created an 18.05 environment by uploading snapshots from `snapshots/18.05`, `Artifact Snapshot` is downloaded into `snapshots/18.06`.
6. Emails the results of recreating old environments to an email address, if specified.

Running the Script

1. Create the `input.properties` file and update it with information for your environment. Save the file in a local, directory. This directory, referred to as `parentsnapshotdirectory` in this discussion. Contents of this file differs depending on your operating system.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run scripts.

2. Create `upgradeSnapshots.ps1` (Windows PowerShell) or `upgradeSnapshots.sh` (Linux/UNIX) script and save it in the `parentsnapshotdirectory` where `input.properties` is located.
3. Create a sub-directory, for example, `snapshots`, within the `parentsnapshotdirectory`.
4. Within the directory that you created in the preceding step (`snapshots`), create a sub-directory for the monthly snapshot that you want to convert to make it compatible with the current EPM Cloud patch level. Name the directory using the `YY.MM` format; for example, `18.05` for the directory to store the May 2018 snapshots.
5. Copy snapshots into the appropriate sub-directory. For example, copy the snapshots for May 2018 into `snapshots/18.05`.
6. Launch the script.
   - Linux/UNIX: run `./upgradeSnapshots.sh`.
   - Windows PowerShell: run `upgradeSnapshots.ps1`.

Windows

Create `input.properties` and `upgradeSnapshots.ps1` script by copying the scripts in this section.

Creating `input.properties`

```
username=exampleAdmin
userpassword=examplePassword
serviceurl=exapleURL
proxyserverusername=proxyServerUserName
proxyserverpassword=proxyPassword
proxyserverdomain=proxyDoamin
parentsnapshotdirectory=C:/some_directory/snapshots
emailtoaddress=exampleAdmin@oracle.com
```

Updating `input.properties`

> **Note:**
>
> If `authentication at proxy server` is not enabled for your Windows network environment, remove the properties `proxyserverusername`, `proxyserverpassword`, and `proxyserverdomain` from the `input.properties` file.

Table 3-8 input.properties Parameters

Parameter	Description
`username`	User name of a Service Administrator.
`userpassword`	Password of the Service Administrator .
`serviceurl`	URL of the environment that is used for this activity.
`proxyserverusername`	The user name to authenticate a secure session with the proxy server that controls access to the internet.
`proxyserverpassword`	The password to authenticate the user with the proxy server.

Table 3-8 (Cont.) input.properties Parameters

Parameter	Description
`proxyserverdomain`	The name of the domain defined for the proxy server.
`parentsnapshotdirectory`	Absolute path of the directory that is to be used as the parent directory of the directory that stores the snapshots to be processed. Use forward slashes (/) as directory separators.
`emailtoaddress`	Optionally, the email address to which the results of recreating old environments are to be sent. The results are emailed only if this value is specified. **Example:** `john.doe@example.com`

> **Note:**
>
> If your password contains special characters, see Handling Special Characters.

Creating `upgradeSnapshots.ps1`

Use this sample script to create `upgradeSnapshots.ps1`

```
# Script for recreating an old EPM Cloud environment

# read in key/value pairs from input.properties file
$inputproperties=ConvertFrom-StringData(Get-Content ./input.properties -raw)

# Global variables
$parentsnapshotdirectory="$($inputproperties.parentsnapshotdirectory)"
$username="$($inputproperties.username)"
$userpassword="$($inputproperties.userpassword)"
$serviceurl="$($inputproperties.serviceurl)"
$proxyserverusername="$($inputproperties.proxyserverusername)"
$proxyserverpassword="$($inputproperties.proxyserverpassword)"
$proxyserverdomain="$($inputproperties.proxyserverdomain)"
$emailtoaddress="$($inputproperties.emailtoaddress)"
$operationmessage="EPM Automate operation:"
$operationfailuremessage="EPM Automate operation failed:"
$operationsuccessmessage="EPM Automate operation completed successfully:"
$epmautomatescript="epmautomate.bat"

$workingdir="$pwd"
$logdir="$workingdir/logs/"
$logfile="$logdir/epmautomate-upgradesnapshots.log"

function LogMessage
{
    $message=$args[0]
    $_mydate=$(get-date -f dd_MM_yy_HH_mm_ss)

    echo "[$_mydate] $message" >> $logfile
}
```

```
function LogAndEchoMessage
{
    $message=$args[0]
    $_mydate=$(get-date -f dd_MM_yy_HH_mm_ss)

    echo "[$_mydate] $message" | Tee-Object -Append -FilePath $logfile
}

function LogOutput
{
    $_mydate=$(get-date -f dd_MM_yy_HH_mm_ss)
    $op=$args[0]
    $opoutput=$args[1]
    $returncode=$args[2]

    #If error
    if ($returncode -ne 0) {
        $failmessage="[$_mydate] $operationfailuremessage $op"
        LogMessage $failmessage
        LogMessage $opoutput
        LogMessage "return code: $returncode"
    } else {
        $successmessage="[$_mydate] $operationsuccessmessage $op"
        LogMessage $successmessage
        LogMessage $opoutput
        LogMessage "return code: $returncode"
    }
}

function ExecuteCommand
{
    $op=$args[0]
    $epmautomatecall="$epmautomatescript $op"
    $date=$(get-date -f dd_MM_yy_HH_mm_ss)

    LogMessage "$operationmessage $epmautomatecall"
    $operationoutput=iex "& $epmautomatecall" >> $logfile 2>&1
    LogOutput $op $operationoutput $LastExitCode
}

function ProcessCommand
{
    $command=$args[0]
    $date=$(get-date -f dd_MM_yy_HH_mm_ss)

    if (!([string]::IsNullOrWhitespace($command))) {
        if (!($command.StartsWith("#"))) {
            ExecuteCommand $command
        }
    }
}

function Init
{
    $logdirexists=Test-Path $logdir
    if (!($logdirexists)) {
```

```
        mkdir $logdir 2>&1 | out-null
    }

    # removing existing epmautomate debug logs
    rm ./*.log

    $logfileexists=Test-Path $logfile
    # remove existing log file
    if ($logfileexists) {
        rm $logfile
    }
}

function GetNextDate
{
    $latestyearmonth=$args[0]
    LogMessage "latest year.month: $latestyearmonth"
    $latestyear,$latestmonth=$latestyearmonth.split('\.')
    LogMessage "latest year: $latestyear"
    LogMessage "latest month: $latestmonth"
    $intlatestyear=[int]$latestyear
    $intlatestmonth=[int]$latestmonth

    if ($intlatestmonth -eq 12) {
        $intnextmonth=1
        $intnextyear=$intlatestyear+1
    } else {
        $intnextmonth=$intlatestmonth+1
        $intnextyear=$intlatestyear
    }

    $nextyear="{0:D2}" -f $intnextyear
    $nextmonth="{0:D2}" -f $intnextmonth

    echo "$nextyear.$nextmonth"
}

function ProcessSnapshot
{
    $snapshotfile=$args[0]
    LogMessage "snapshotfile: $snapshotfile"
    $nextdate=$args[1]
    LogMessage "nextdate: $nextdate"
    $snapshotfilename=$snapshotfile.split('/')[-1]
    LogMessage "snapshotfilename: $snapshotfilename"
    $snapshotname=$snapshotfilename.split('.')[0]
    LogMessage "snapshotname: $snapshotname"

    ProcessCommand
"login $username $userpassword $serviceurl $proxyserverusername $proxyserverpa
ssword $proxyserverdomain"
    ProcessCommand "recreate -f"
    ProcessCommand "uploadfile $snapshotfile"
    ProcessCommand "importsnapshot $snapshotname"
    ProcessCommand "runDailyMaintenance skipNext=true -f"
    ProcessCommand "downloadfile 'Artifact Snapshot'"
```

```
    ProcessCommand "deletefile $snapshotname"
    ProcessCommand "logout"

    $nextdatedirexists=Test-Path $parentsnapshotdirectory/$nextdate
    if (!($nextdatedirexists)) {
        mkdir $parentsnapshotdirectory/$nextdate 2>&1 | out-null
    }

    LogMessage "Renaming 'Artifact Snapshot.zip' to $snapshotname.zip and
moving to $parentsnapshotdirectory/$nextdate"
    mv $workingdir/'Artifact Snapshot.zip' $workingdir/$snapshotname.zip
>> $logfile 2>&1
    mv $workingdir/$snapshotname.zip $parentsnapshotdirectory/$nextdate
>> $logfile 2>&1
}

function callSendMail
{
    $logfile=$logfile -replace "\\", "/"
    $elements=$logfile.split('/')
    $logfilename=$elements[-1]

    if (${emailtoaddress} -match "@") {
        epmautomate.bat login ${username} ${userpassword} ${serviceurl}
        epmautomate.bat uploadFile "$logfile"
        epmautomate.bat sendMail $emailtoaddress "Recreating An Old EPM Cloud
Environment results" Body="The results of recreating an old EPM Cloud
Environment are attached." Attachments=$logfilename
        epmautomate.bat deleteFile "$logfilename"
        epmautomate.bat logout
    }
}

#----- main body of processing
date
Init
LogAndEchoMessage "Starting upgrade snapshots processing"
$snapshotdirs=@(Get-ChildItem -Directory "$parentsnapshotdirectory" -name)
LogMessage "snapshot directories: $snapshotdirs"
$latestreleasedate=$snapshotdirs[-1]
LogMessage "latest release date: $latestreleasedate"
$latestreleasesnapshotdir="$parentsnapshotdirectory/$latestreleasedate"
LogMessage "latest release snapshot dir: $latestreleasesnapshotdir"
$nextdate=$(GetNextDate "$latestreleasedate")
$snapshotfiles=@(Get-ChildItem -File "$latestreleasesnapshotdir")
if ($snapshotfiles.length -eq 0) {
    LogAndEchoMessage "No snapshot files found in
directory $latestreleasesnapshotdir. Exiting script."
    exit
}
foreach ($snapshotfile in $snapshotfiles) {
    LogAndEchoMessage "Processing snapshotfile: $snapshotfile"
    ProcessSnapshot $latestreleasesnapshotdir/$snapshotfile $nextdate
}
LogAndEchoMessage "Upgrade snapshots processing completed"
```

```
date
callSendMail
```

Linux/UNIX

Create `upgradeSnapshots.sh` and `input.properties` by copying the following scripts.

Creating `input.properties` for Linux/UNIX

> **Note:**
>
> If your network is not configured to use a proxy server to access the internet, remove the properties `proxyserverusername`, `proxyserverpassword`, and `proxyserverdomain` from the `input.properties` file.

```
username=exampleAdmin
userpassword=examplePassword
serviceurl=exapleURL
proxyserverusername=
proxyserverpassword=
proxyserverdomain=
jdkdir=/home/user1/jdk160_35
epmautomatescript=/home/exampleAdmin/epmautomate/bin/epmautomate.sh
parentsnapshotdirectory=/home/exampleAdmin/some_directory/snapshots
emailtoaddress=exampleAdmin@oracle.com
```

Updating `input.properties`

Table 3-9 input.properties Parameters

Parameter	Description
`username`	User name of a Service Administrator.
`userpassword`	Password of the Service Administrator .
`serviceurl`	URL of the environment that is being used for this activity.
`proxyserverusername`	The user name to authenticate a secure session with the proxy server that controls access to the internet.
`proxyserverpassword`	The password to authenticate the user with the proxy server.
`proxyserverdomain`	The name of the domain defined for the proxy server.
`jdkdir`	`JAVA_HOME` location.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`).
`parentsnapshotdirectory`	Absolute path of the directory that is to be used as the parent directory of the directory that stores the snapshot to be processed.
`emailtoaddress`	Optionally, the email address to which the results of recreating old environments are to be sent.

> **Note:**
>
> If your password contains special characters, see Handling Special Characters.

Creating `upgradeSnapshots.sh`

Use this sample script to create `upgradeSnapshots.sh`

```
#!/bin/sh

. ./input.properties
workingdir=$(pwd)
logdir="${workingdir}/logs"
logfile=epmautomate-upgradesnapshots.log
operationmessage="EPM Automate operation:"
operationfailuremessage="EPM Automate operation failed:"
operationsuccessmessage="EPM Automate operation completed successfully:"
logdebugmessages=true

if [ ! -d ${jdkdir} ]
then
    echo "Could not locate JDK/JRE. Please set value for "jdkdir" property in
input.properties file to a valid JDK/JRE location."
    exit
fi

if [ ! -f ${epmautomatescript} ]
then
    echo "Could not locate EPM Automate script. Please set value for
"epmautomatescript" property in the input.properties file."
    exit
fi

export JAVA_HOME=${jdkdir}

debugmessage() {
    # logdebugmessages is defined (or not) in testbase input.properties
    if [ "${logdebugmessages}" = "true" ]
    then
        logmessage "$1"
    fi
}

logmessage()
{
    local message=$1
    local _mydate=$(date)

    echo "[$_mydate] ${message}" >> "$logdir/$logfile"
}

echoandlogmessage()
{
    local message=$1
```

```
    local _mydate=$(date)

    echo "[$_mydate] ${message}" | tee -a ${logdir}/${logfile}
}

logoutput()
{
    date=`date`
    op="$1"
    opoutput="$2"
    returncode="$3"

    #If error
    #if grep -q "EPMAT-" <<< "$2"
    if [ $returncode -ne 0 ]
    then
        failmessage="[${date}] ${operationfailuremessage} ${op}"
        logmessage "${failmessage}"
        logmessage "${opoutput}"
        logmessage "return code: ${returncode}"
    else
        successmessage="${operationsuccessmessage} ${op}"
        logmessage "${successmessage}"
        logmessage "${opoutput}"
        logmessage "return code: ${returncode}"
    fi
}

getLatestReleaseSnapshotDir()
{
    local snapshotdirs=$(find ${parentsnapshotdirectory} -type d | sort)
    debugmessage "snapshot directories: ${snapshotdirs}"
    local latestreleasesnapshotdir=$(echo ${snapshotdirs##*$\n} | rev | cut -
d' ' -f1 | rev)
    debugmessage "latest release snapshot dir: ${latestreleasesnapshotdir}"
    echo "${latestreleasesnapshotdir}"
}

getNextDate()
{
    local thisyearmonth=$1
    local thisyear=$(echo ${thisyearmonth} | cut -d'.' -f1)
    local thismonth=$(echo ${thisyearmonth} | cut -d'.' -f2)

    intthismonth=$(bc <<< ${thismonth})
    intthisyear=$(bc <<< ${thisyear})

    if [ ${intthismonth} -eq 12 ]
    then
        local intnextmonth=1
        local intnextyear=$((intthisyear+1))
    else
        local intnextmonth=$((intthismonth+1))
        local intnextyear=${intthisyear}
    fi
```

```
    nextmonth=$(printf "%02d\n" ${intnextmonth})
    nextyear=$(printf "%02d\n" ${intnextyear})

    debugmessage "next date: ${nextyear}.${nextmonth}"

    echo "${nextyear}.${nextmonth}"
}

init()
{
    if [ ! -d "$logdir" ]
    then
        mkdir $logdir
    fi

    # removing existing epmautomate debug logs
    if ls ./*.log >/dev/null 2>&1
    then
       rm ./*.log
    fi

    # remove existing log files
    if [ -f "${logdir}/${logfile}" ]
    then
        rm ${logdir}/${logfile}
    fi
}

processCommand()
{
    op="$1"
    date=`date`

    logmessage "$operationmessage $op"
    operationoutput=`eval "$epmautomatescript $op"`
    logoutput "$op" "$operationoutput" "$?"
}

processSnapshot()
{
    local snapshotfile="$1"
    local nextdate="$2"
    local snapshotname=$(echo "${snapshotfile}" | rev | cut -d'/' -f1 | rev |
cut -d'.' -f1)

    processCommand "login ${username} ${userpassword} ${serviceurl} $
{proxyserverusername} ${proxyserverpassword}"
    processCommand "recreate -f"
    processCommand "uploadfile ${snapshotfile}"
    processCommand "importsnapshot \"${snapshotname}\""
    processCommand "runDailyMaintenance skipNext=true -f"
    processCommand "downloadfile \"Artifact Snapshot\""
    processCommand "deletefile \"${snapshotname}\""
    processCommand "logout"

    if [ ! -d ${parentsnapshotdirectory}/${nextdate} ]
```

```
    then
        mkdir ${parentsnapshotdirectory}/${nextdate}
    fi
runDailyMaintenance -f
    logmessage "Renaming \"Artifact Snapshot.zip\" to ${snapshotname}.zip and
moving to ${parentsnapshotdirectory}/${nextdate}"
    mv "${workingdir}/Artifact Snapshot.zip" "${workingdir}/$
{snapshotname}.zip" >> "$logdir/$logfile" 2>&1
    mv "${workingdir}/${snapshotname}.zip" ${parentsnapshotdirectory}/$
{nextdate} >> "$logdir/$logfile" 2>&1
}

callSendMail() {

    if [[ "${emailtoaddress}" == *"@"* ]]
    then
        ${epmautomatescript} login ${username} ${userpassword} ${serviceurl}
        ${epmautomatescript} uploadFile "$logdir/$logfile"
        ${epmautomatescript} sendMail $emailtoaddress "Recreating An Old EPM
Cloud Environment results" Body="The results of recreating an old EPM Cloud
Environment are attached" Attachments=$logfile
        ${epmautomatescript} deleteFile "$logfile"
        ${epmautomatescript} logout
    fi
}

#----- main body of processing
date
echoandlogmessage "Starting upgrade snapshots processing"
init
latestreleasesnapshotdir=$(getLatestReleaseSnapshotDir)
latestreleasedate=$(echo "${latestreleasesnapshotdir}" | rev | cut -d'/' -f1
| rev)
debugmessage "latest release date: ${latestreleasedate}"
nextdate=$(getNextDate ${latestreleasedate})

snapshotfiles=$(find ${latestreleasesnapshotdir} -type f -name \*.zip | tr
"\n" "|")
if [ ${#snapshotfiles} -eq 0 ]
then
    echoandlogmessage "No snapshot files found in directory $
{latestreleasesnapshotdir}"
fi

IFS="|"
for snapshotfile in $snapshotfiles
do
    echoandlogmessage "Processing snapshotfile: ${snapshotfile}"
    processSnapshot ${snapshotfile} ${nextdate}
done
unset IFS
echoandlogmessage "Upgrade snapshots processing completed."
callSendMail
```

Automate Database Access Audit and Compliance

Use the PowerShell and Bash Shell scripts in this section to leverage EPM Automate commands to collect audit and compliance data around manual database access.

You can use these scripts to complete these tasks:

- Download the Activity Report for the current day
- Parse the report to determine if manual database access is reported for the environment
- Create `./reports/dataAccessAuditReport.txt` relative to the directory from where you execute the script. The report lists the time of database access and the SQL command that was executed. This is a cumulative file, which shows the latest information at the top. Information available include:
  - Date and time at which the report was generated
  - Database access details, if available. Database access without a service request and database access with service request are listed in separate sections.
    If manual database access is not reported in the Activity Report, the report states `No SQL statements executed`.
  - Optionally, send the report to a specified email address.

To automate data access audit and compliance:

1. Copy one of the script from the following sections to a file and save it to your file system. Name the file `parseActivityReport.ps1` (Windows see PowerShell Script (parseActivityReport.ps1)) or `parseActivityReport.sh` (Linux/UNIX see Bash Shell Script (parseActivityReport.sh)).
2. **Windows only:** Create a batch file named `parseActivityReport.bat` by copying the following script into a file. Save the file in the directory where `parseActivityReport.ps1` is stored.

```
@echo off
set paramRequiredMessage=Syntax: parseActivityReport.bat USERNAME PASSWORD/
PASSWORD_FILE URL [REPORT_EMAIL_TO_ADDRESS]

if "%~1" == "" (
  echo User Name is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~2" == "" (
  echo Password or Password_File is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )
if "%~3" == "" (
  echo URL is missing.
  echo %paramRequiredMessage%
  exit /b 1
  )

PowerShell.exe -File parseActivityReport.ps1 %*
```

3. Modify `parseActivityReport.bat` (Windows) or `parseActivityReport.sh` (Linux/UNIX see) to set the values for the parameters in the following table.

Table 3-10 Variable Values to Include in Scripts

Variable	Description
`epmuser`	User name of a Service Administrator **Examples:** Windows: `set epmuser="jDoe"` Linux/UNIX: `epmuser="jDoe"`
`epmpassword`	Password of the Service Administrator or the location of the encrypted password file. See the encrypt command for information on creating an encrypted password file. If your password contains special characters, see Handling Special Characters. **Examples:** Windows: `set epmpassword = "Example"` Linux/UNIX: `epmpassword="Example"`
`epmurl`	The URL of the Oracle Enterprise Performance Management Cloud environment. **Examples:** Windows: `set epmurl="https://example.oraclecloud.com"` Linux/UNIX: `epmurl="https://example.oraclecloud.com"`
`report_email_to_address`	Optionally, the email address to which the report is to be sent. The report is emailed only if this value is specified. **Example: `john.doe@example.com`**

4. **For `parseActivityReport.sh` only:** Ensure that the following values are set correctly for your system:
   - *JAVA_HOME*
   - Location of `epmautomatescript.sh` by updating the value of `epmautomatescript` directive
5. Using a scheduler available on the operating system, schedule `parseActivityReport.bat` (which executes `parseActivityReport.ps1`) or `parseActivityReport.sh` to run once every day. See Automating Script Execution.

PowerShell Script (parseActivityReport.ps1)

```
# Parse Activity Report script

$epmuser=$args[0]
$epmpassword=$args[1]
$epmurl=$args[2]
$reportemailtoaddress=$args[3]

$logdir="./logs"
$logfile="${logdir}/data_access.log"
$reportdir="./reports"
$reportfile="${reportdir}/dataAccessAuditReport.txt"
$matchfile="${reportdir}/matchfile.txt"
$nosrfile="${reportdir}/data_access_nosr.csv"
```

```
$srfile="${reportdir}/data_access_sr.csv"
$aprfilelist="${reportdir}/aprfilelist.txt"
$activityreportfilelist="${reportdir}/activityreportfiles.txt"
$activityreportregex='apr/[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}_[0-9]{2}_[0-9]
{2}/[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}_[0-9]{2}_[0-9]{2}.html'

$global:activityreportfile=""

$NO_SQL_EXECUTED_STATEMENT="No SQL statements executed"
$SQL_WITH_SR_EXECUTED_STATEMENT="SQL statements executed with an SR"
$SQL_WITH_NO_SR_EXECUTED_STATEMENT="SQL statements executed without an SR"

function DownloadLatestActivityReport() {
    epmautomate.bat login ${epmuser} ${epmpassword} ${epmurl} >> ${logfile}
    epmautomate.bat listfiles > ${aprfilelist}
    foreach ($line in Get-Content $aprfilelist) {
        if ($line -match $activityreportregex){
            echo "$line" >> $activityreportfilelist
        }
    }
    $global:activityreportfile=Get-Content ${activityreportfilelist} -Tail 1
    $global:activityreportfile=$global:activityreportfile.trim()
    echo " "
    echo "Processing activity report file: $global:activityreportfile" | tee -
a ${logfile}
    epmautomate.bat downloadfile "$global:activityreportfile" >> ${logfile}
    epmautomate.bat logout >> ${logfile}
}

function deleteLine($file, $start, $end) {
    $i = 0
    $start--
    $end--
    (Get-Content $file) | where{
        ($i -lt $start -or $i -gt $end)
        $i++
    } > $file
    #(Get-Content $file)
}

function GenerateCsvs()
{
    $sqlregex='<DIV id="Database">.*?</DIV>'
    $activityreportfilename=Split-Path $global:activityreportfile -leaf

    echo "Creating CSV file: ${matchfile} from data in activityreportfile: $
{activityreportfilename}" >> ${logfile}
    # remove tab and newline characters
    $activityreportexists=Test-Path "$activityreportfilename"
    if ($activityreportexists) {
        (Get-Content "$activityreportfilename") -join ' ' | Set-Content
"$activityreportfilename"
        (Get-Content "$activityreportfilename") -replace "`t", "" | Set-
Content "$activityreportfilename"
    }
```

```
    # capture text matching regex
    $string=Get-Content $activityreportfilename
    $ans=$string -match $sqlregex

    if ($ans -eq "True") {
        $Matches.0 > $matchfile
        # remove HTML tags, etc.
        (Get-Content "$matchfile") -replace "<tr", "`n<tr" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "<tr[^>]*>", "" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "<th[^>]*>", "" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "<td[^>]*>", "|" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "<br>", "" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "</td>", "" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "</tr>", "" | Set-Content
"$matchfile"
        (Get-Content "$matchfile") -replace "\s*</table>\s*</DIV>", "" | Set-
Content "$matchfile"
        deleteLine $matchfile 1 2

        # create SR, NOSR CSV files
        Get-Content $matchfile | ForEach-Object {
            $elements=$_.split('|')
            $timeval=$elements[1].Trim()
            $srval=$elements[3].Trim()
            $sqlval=$elements[4].Trim()

            if (${srval} -eq "") {
                echo "${timeval}|${sqlval}" >> ${nosrfile}
            } else {
                if (${sqlval} -ne "") {
                    echo "${srval}|${timeval}|${sqlval}" >> ${srfile}
                }
            }
        }

    } else { # no SQL statements in activity report
        echo "" >> ${reportfile}
        echo $(date) >> ${reportfile}
        echo "Processing activity report file: $global:activityreportfile"
>> ${reportfile}
        echo "${NO_SQL_EXECUTED_STATEMENT}" | tee -a ${reportfile}
        CleanUp
        EmailReportResults
        exit
    }
}

function ReportResults() {
    echo $(date) >> ${reportfile}
    echo "Processing activity report file: $global:activityreportfile" >> $
```

```
{reportfile}
    $srfileexists=Test-Path $srfile
    if ($srfileexists) {
        echo "" | tee -a ${reportfile}
        echo "${SQL_WITH_SR_EXECUTED_STATEMENT}" | tee -a ${reportfile}
        echo "SR#        Time        SQL Statement" | tee -a ${reportfile}
        echo "---        ----        -------------" | tee -a ${reportfile}

        # Loop through csv file and parse
        Get-Content $srfile | ForEach-Object {
            $elements=$_.split('|')
            $srval=$elements[0]
            $timeval=$elements[1]
            $sqlval=$elements[2]
            echo "${srval}    ${timeval}    ${sqlval}" | tee -a ${reportfile}
        }
    }

    $nosrfileexists=Test-Path $nosrfile
    if ($nosrfileexists) {
        echo "" | tee -a ${reportfile}
        echo "${SQL_WITH_NO_SR_EXECUTED_STATEMENT}" | tee -a ${reportfile}
        echo "Time        SQL Statement" | tee -a ${reportfile}
        echo "----        -------------" | tee -a ${reportfile}

        # Loop through csv file and parse
        Get-Content $nosrfile | ForEach-Object {
            $elements=$_.split('|')
            $timeval=$elements[0]
            $sqlval=$elements[1]
            echo "${timeval}    ${sqlval}" | tee -a ${reportfile}
        }
    }
    EmailReportResults
}

function EmailReportResults
{
  $elements=$reportfile.split('/')
  $reportfilename=$elements[2]

  if (${reportemailtoaddress} -match "@") {
    echo "Emailing Activity Report Results" | tee -a ${logfile}
    epmautomate.bat login ${epmuser} ${epmpassword} ${epmurl} >> ${logfile}
    epmautomate.bat uploadFile $reportfile >> ${logfile}
    epmautomate.bat sendMail $reportemailtoaddress "Database Access Audit
Report Results" Body="Database Access Audit Report Results are attached."
Attachments=$reportfilename >> ${logfile}
    epmautomate.bat deleteFile $reportfilename >> ${logfile}
    epmautomate.bat logout >> ${logfile}
  }
}

function Init
{
    $logdirexists=Test-Path $logdir
```

```
    if (!($logdirexists)) {
        mkdir $logdir 2>&1 | out-null
    }

    $reportdirexists=Test-Path $reportdir
    if (!($reportdirexists)) {
        mkdir $reportdir 2>&1 | out-null
    }

    $logfileexists=Test-Path $logfile
    if ($logfileexists) {
        rm $logfile 2>&1 | out-null
    }

    $matchfileexists=Test-Path $matchfile
    if ($matchfileexists) {
        rm $matchfile 2>&1 | out-null
    }

    $nosrfileexists=Test-Path $nosrfile
    if ($nosrfileexists) {
        rm $nosrfile 2>&1 | out-null
    }

    $srfileexists=Test-Path $srfile
    if ($srfileexists) {
        rm $srfile 2>&1 | out-null
    }

    $aprfilelistexists=Test-Path $aprfilelist
    if ($aprfilelistexists) {
        rm $aprfilelist 2>&1 | out-null
    }

    $activityreportfilelistexists=Test-Path $activityreportfilelist
    if ($activityreportfilelistexists) {
        rm $activityreportfilelist 2>&1 | out-null
    }
}

function CleanUp
{
    $matchfileexists=Test-Path $matchfile
    if ($matchfileexists) {
        rm $matchfile 2>&1 | out-null
    }

    $aprfilelistexists=Test-Path $aprfilelist
    if ($aprfilelistexists) {
        rm $aprfilelist 2>&1 | out-null
    }

    $activityreportfilelistexists=Test-Path $activityreportfilelist
    if ($activityreportfilelistexists) {
        rm $activityreportfilelist 2>&1 | out-null
    }
```

```
}

Init
DownloadLatestActivityReport
GenerateCsvs
ReportResults
CleanUp
```

Bash Shell Script (parseActivityReport.sh)

```
#!/bin/sh

export JAVA_HOME=/scratch/dteHome/autoWork/jdk1.8.0_191
epmautomatescript=/scratch/dteHome/autoWork/epmautomate/19.11.55/bin/
epmautomate.sh

epmuser="<EPM USER>"
epmpwd="<EPM PASSWORD>"
epmurl="<EPM URL>"
reportemailtoaddress="<EMAIL ADDRESS>"

logdir=./logs
logfile="${logdir}/data_access.log"
reportdir=./reports
reportfile="${reportdir}/dataAccessAuditReport.txt"
nosrfile="${reportdir}/data_access_nosr.csv"
srfile="${reportdir}/data_access_sr.csv"
matchfile="${reportdir}/match.out"
aprfilelist="${reportdir}/aprfilelist.txt"
activityreportfile=""
activityreportregex='apr/[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}_[0-9]{2}_[0-9]
{2}/[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}_[0-9]{2}_[0-9]{2}.html'

NO_SQL_EXECUTED_STATEMENT="No SQL statements executed".
SQL_WITH_SR_EXECUTED_STATEMENT="SQL statements executed with an SR"
SQL_WITH_NO_SR_EXECUTED_STATEMENT="SQL statements executed without an SR"

cd "$(dirname "$0")"

generateCsvs()
{
    local sqlregex='<DIV id="Database">.*?</DIV>'
    local activityreportfilename=$(echo "${activityreportfile}" | rev | cut -
d'/' -f1 | rev)

    echo "Creating CSV file: ${matchfile} from data in activityreportfile: $
{activityreportfilename}" >> ${logfile}
    # remove tab and newline characters
    cat "${activityreportfilename}" | tr -d "\t\n\r" > ${matchfile}
    # capture text matching regex
    grep -Po "${sqlregex}" ${matchfile} > ${matchfile}.tmp

    # remove HTML tags, etc.
    sed -e 's/<tr/\n<tr/g' -e 's/<tr[^>]*>//g' -e 's/<th[^>]*>//g' -e 's/
<td[^>]*>/|/g' -e 's/<br>//g' -e 's|</td>||g' -e 's|</tr>||g' -e 's|[ ]*</
```

```
table></DIV>||g' -e 's/|[ ]*/|/g' -e 's/[ ]*|/|/g' -e 's/<DIV
id="Database">.*<!-- Print Tables -->\n//g' ${matchfile}.tmp > ${matchfile}

    # create SR, NOSR CSV files
    while read line
    do
        timeval=$(echo "${line}" | cut -d'|' -f2)
        srval=$(echo "${line}" | cut -d'|' -f4)
        sqlval=$(echo "${line}" | cut -d'|' -f5)

        if [[ "${srval}" == "" ]]
        then
            echo "${timeval}|${sqlval}" >> ${nosrfile}
        else
            if [[ "${sqlval}" != "" ]]
            then
                echo "${srval}|${timeval}|${sqlval}" >> ${srfile}
            fi
        fi
    done < ${matchfile}
}

reportResults() {
    echo $(date) >> ${reportfile}
    echo "Processing activity report file: $activityreportfile" >> $
{reportfile}
    if [[ -f ${srfile} ]]
    then
        echo "" | tee -a ${reportfile}
        echo "${SQL_WITH_SR_EXECUTED_STATEMENT}" | tee -a ${reportfile}
        echo "SR#        Time        SQL Statement" | tee -a ${reportfile}
        echo "---        ----        -------------" | tee -a ${reportfile}
        while read line
        do
            srval=$(echo "${line}" | cut -d'|' -f1)
            timeval=$(echo "${line}" | cut -d'|' -f2)
            sqlval=$(echo "${line}" | cut -d'|' -f3)
            echo "${srval}    ${timeval}    ${sqlval}" | tee -a ${reportfile}
        done < ${srfile}
    fi

    if [[ -f ${nosrfile} ]]
    then
        echo "" | tee -a ${reportfile}
        echo "${SQL_WITH_NO_SR_EXECUTED_STATEMENT}" | tee -a ${reportfile}
        echo "Time        SQL Statement" | tee -a ${reportfile}
        echo "----        --- ---------" | tee -a ${reportfile}
        while read line
        do
            timeval=$(echo "${line}" | cut -d'|' -f1)
            sqlval=$(echo "${line}" | cut -d'|' -f2)
            echo "${timeval}    ${sqlval}" | tee -a ${reportfile}
        done < ${nosrfile}
    fi

    if [[ ! -f ${srfile} ]] && [[ ! -f ${nosrfile} ]]
```

```
    then
        echo "" | tee -a ${reportfile}
        echo "${NO_SQL_EXECUTED_STATEMENT}" | tee -a ${reportfile}
    fi

    emailReportResults
}

downloadLatestActivityReport() {
    ${epmautomatescript} login ${epmuser} ${epmpwd} ${epmurl} >> ${logfile}
    ${epmautomatescript} listfiles > ${aprfilelist}
    activityreportfile=$(cat ${aprfilelist} | grep -P "$
{activityreportregex}" | tail -n 1 | sed -e 's/^ //')
    echo " "
    echo "Processing activity report file: ${activityreportfile}" | tee -a $
{logfile}
    ${epmautomatescript} downloadfile "${activityreportfile}" >> ${logfile}
    ${epmautomatescript} logout >> ${logfile}
}

emailReportResults() {
    reportfilename=$(echo "${reportfile}" | cut -d'/' -f3)

    if [[ "${reportemailtoaddress}" == *"@"* ]]
    then
        echo "Emailing Activity Report Results" | tee -a ${logfile}
        ${epmautomatescript} login ${epmuser} ${epmpwd} ${epmurl} >> $
{logfile}
        ${epmautomatescript} uploadFile "$reportfile" >> ${logfile}
        ${epmautomatescript} sendMail $reportemailtoaddress "Database Access
Audit Report Results" Body="Database Access Audit Report Results are
attached." Attachments=$reportfilename >> ${logfile}
        ${epmautomatescript} deleteFile "$reportfilename" >> ${logfile}
        ${epmautomatescript} logout >> ${logfile}
    fi
}

checkParams()
{
    if [ -z "$epmuser" ]
    then
        echo "Username is missing."
        echo "Syntax: parseActivityReport.sh USERNAME PASSWORD URL"
        exit 2
    fi

    if [ -z "$epmpwd" ]
    then
        echo "Password is missing."
        echo "Syntax: parseActivityReport.sh USERNAME PASSWORD URL"
        exit 2
    fi

    if [ -z "$epmurl" ]
    then
        echo "URL is missing."
```

```
        echo "Syntax: parseActivityReport.sh USERNAME PASSWORD URL"
        exit 2
    fi
}

init()
{
    checkParams

    if [ ! -d "${logdir}" ]
    then
        mkdir ${logdir}
    fi

    if [ ! -d "${reportdir}" ]
    then
        mkdir ${reportdir}
    fi

    if [ ! -f "${epmautomatescript}" ]
    then
        echo "Cannot locate EPMAutomate script: ${epmautomatescript}. Please
check setting and run script again. Exiting." | tee -a ${logfile}
        exit
    fi

    if [ -f "${srfile}" ]
    then
        rm ${srfile}
    fi

    if [ -f "${nosrfile}" ]
    then
        rm ${nosrfile}
    fi

    if [ -f "${matchfile}" ]
    then
        rm ${matchfile}
    fi

    if [ -f "${aprfilelist}" ]
    then
        rm ${aprfilelist}
    fi
}

cleanup()
{
    if [ -f "${matchfile}" ]
    then
        rm ${matchfile}
    fi

    if [ -f "${matchfile}.tmp" ]
    then
```

```
        rm ${matchfile}.tmp
    fi

    if [ -f "${aprfilelist}" ]
    then
        rm ${aprfilelist}
    fi
}

init
downloadLatestActivityReport
generateCsvs
reportResults
cleanup
```

Replicate Users and Predefined Role Assignments

The scripts in this section helps you migrate users and predefined role assignments of an environment to another.

About the Scripts

You use two distinct scripts: one to replicate users across identity domains and another to replicate predefined role assignments of the users. The order for running these scripts is as follows:

- Run the script for replicating users (`replicateusers`) and verify that all users are created in the target identity domain. The user running this script must have the Identity Domain Administrator and Service Administrator roles in both environments.
- Run the script for replicating role assignments (`replicatepredefinedroles`).

> **Note:**
>
> - If the passwords contain special characters, see Handling Special Characters
> - The scripts in this section work only for predefined roles: Service Administrator, Power User, User, and Viewer.

Running the Scripts

For information on creating the required scrips and batch files, see these topics:

- Replicating the Users of One Identity Domain in Another
- Replicating Predefined Role Assignments from One Environment to Another

Windows Steps

1. Create `replicateusers.bat`, `replicateusers.ps1`, `replicatepredefinedroles.bat`, and `replicatepredefinedroles.ps1` and save them in a local directory in which you have write and execute privileges.
2. Update the batch files with information for the source and target environments, and internet proxy server, if needed.

3. Run `replicateusers.bat`, which executes `replicateusers.ps1`. You must specify the default password to be assigned to replicated users as a command line parameter as follows:
   `replicateusers.bat Pwd_for_users`

   If the password contains special characters, be sure to use the appropriate escape character. See Handling Special Characters.

4. Run `replicatepredefinedroles.bat` to create role assignments identical to those that exist in the source environment.

Linux/UNIX Steps

1. Create the `replicateusers.sh` and `replicatepredefinedroles.sh` scripts and save them in a local directory in which you have write and execute privileges.
2. Update `replicateusers.sh` and `replicatepredefinedroles.sh` with information for the source and target environments, and internet proxy server, if needed.
3. Run `replicateusers.sh`. You must specify the default password to be assigned to replicated users as a command line parameter as follows:
   `./replicateusers.sh Pwd_for_users`

   If the password contains special characters, be sure to use the appropriate escape character. See Handling Special Characters.

4. Run `replicatepredefinedroles.sh` script to create role assignments identical to those that exist in the source environment.

Replicating the Users of One Identity Domain in Another

Use the scripts in this section to clone users of one identity domain to another identity domain. The user running these scripts must have the Identity Domain Administrator and Service Administrator roles in the source and target environments.

Windows

Create `replicateusers.bat` and `replicateusers.ps1` by copying the scripts in this section.

1. Create `replicateusers.ps1` by copying this script:

```
# Replicate users script

param(
  [string]$epmusersource,
  [string]$epmpwdsource,
  [string]$epmurlsource,
  [string]$epmidentitydomainsource,
  [string]$epmusertarget,
  [string]$epmpwdtarget,
  [string]$epmurltarget,
  [string]$epmidentitydomaintarget,
  [string]$proxyserverusername,
  [string]$proxyserverpassword,
  [string]$proxyserverdomain,
  [string]$userpassword,
  [string]$resetpassword,
  [string]$emailtoaddress
)
```

```
$roleassignmentreport="roleassignmentreport.csv"
$usersreport="users.csv"

echo "Replicate users script started"


# delete existing reports
$roleassignmentreportexists=Test-Path $roleassignmentreport
if ($roleassignmentreportexists) {
    rm $roleassignmentreport 2>&1 | out-null
}

$usersreportexists=Test-Path $usersreport
if ($usersreportexists) {
    rm $usersreport 2>&1 | out-null
}

# epmautomate login Source App as an IDM Admin
echo "Logging into source application at ${epmurlsource}"
epmautomate login ${epmusersource} ${epmpwdsource} ${epmurlsource} $
{epmidentitydomainsource} ${proxyserverusername} ${proxyserverpassword} $
{proxyserverdomain}
echo "Creating role assignment report: ${roleassignmentreport}"
epmautomate roleAssignmentReport ${roleassignmentreport}
if (${emailtoaddress} -match "@") {
    epmautomate.bat sendMail $emailtoaddress "Role assignment report"
Body="Role assignment report is attached."
Attachments=$roleassignmentreport}
echo "Downloading role assignment report"
epmautomate downloadfile ${roleassignmentreport}
epmautomate deletefile ${roleassignmentreport}
epmautomate logout

# Create users report
Get-Content ${roleassignmentreport} | ForEach-Object {
    $user=$_.split(',')[0]
    $firstname=$_.split(',')[1]
    $lastname=$_.split(',')[2]
    $email=$_.split(',')[3]

    if ($firstname -eq "First Name") {
        return
    } else {
        echo "${firstname},${lastname},${email},${user}" >> ${usersreport}
    }

}

Get-Content -Path "${usersreport}" | Sort-Object -Unique > "$
{usersreport}.tmp"
mv -Force "${usersreport}.tmp" "${usersreport}"
$userheader="First Name,Last Name,Email,User Login"
"${userheader}`r`n" + (Get-Content $usersreport -Raw) | Set-
Content $usersreport

# epmautomate login Target App as an IDM Admin
```

```
echo "Logging into target application at ${epmurltarget}"
epmautomate login ${epmusertarget} ${epmpwdtarget} ${epmurltarget} $
{epmidentitydomaintarget} ${proxyserverusername} ${proxyserverpassword} $
{proxyserverdomain}
epmautomate deletefile ${usersreport} | Out-Null
echo "Uploading file ${usersreport}"
epmautomate uploadfile ${usersreport}
echo "Adding users"
epmautomate addUsers ${usersreport} userPassword=${userpassword}
resetPassword=${resetpassword}
epmautomate deletefile ${usersreport}
epmautomate logout
rm deletefile*.log | Out-Null
echo "Replicate users script completed"
```

2. Create `replicateusers.bat` by copying this script:

```
@ECHO OFF
SET thisdir=%~dp0
SET scriptpath=%thisdir%replicateusers.ps1
SET paramRequiredMessage=Syntax: replicateusers.bat "USER_PASSWORD"

REM USER DEFINED VARIABLES
REM ------------------------
set epmusersource="<EPM USER FOR SOURCE ENVIRONMENT>"
set epmpwdsource="<EPM PASSWORD FOR SOURCE ENVIRONMENT>"
set epmurlsource="<EPM URL FOR SOURCE ENVIRONMENT>"
set epmidentitydomainsource="<EPM IDENTITY DOMAIN FOR SOURCE ENVIRONMENT>"
set epmusertarget="<EPM USER FOR TARGET ENVIRONMENT>"
set epmpwdtarget="<EPM PASSWORD FOR TARGET ENVIRONMENT>"
set epmurltarget="<EPM URL FOR TARGET ENVIRONMENT>"
set epmidentitydomaintarget="<EPM IDENTITY DOMAIN FOR TARGET ENVIRONMENT>"
set proxyserverusername="<PROXY SERVER USER NAME>"
set proxyserverpassword="<PROXY SERVER PASSWORD>"
set proxyserverdomain="<PROXY SERVER DOMAIN>"
set resetpassword=false
set emailtoaddress="<EMAIL_TO_ADDRESS>"
REM ------------------------

if "%~1" == "" (
        echo USER_PASSWORD is missing. This is used to set the default
password for the replicated users.
        echo %paramRequiredMessage%
        exit /b 1
  )

PowerShell -NoProfile -ExecutionPolicy Bypass -Command "& '%scriptpath%' -
epmusersource '%epmusersource%' -epmpwdsource '%epmpwdsource%' -
epmurlsource '%epmurlsource%' -epmidentitydomainsource
'%epmidentitydomainsource%' -epmusertarget '%epmusertarget%' -epmpwdtarget
'%epmpwdtarget%' -epmurltarget '%epmurltarget%' -epmidentitydomaintarget
'%epmidentitydomaintarget%' -proxyserverusername '%proxyserverusername%' -
proxyserverpassword '%proxyserverpassword%' -proxyserverdomain
'%proxyserverdomain%' -userpassword '%~1' -resetpassword '%resetpassword%'
-emailtoaddress '%emailtoaddress%'"
```

3. Update `replicateusers.bat`. See the following table for the values you must specify.

Parameter	Description
`epmusersource`	User name of a user with Identity Domain Administrator and Service Administrator roles in the source environment. **Examples:** Windows: `set epmusersource="jDoe"` Linux/UNIX: `epmusersource="jDoe"`
`epmpwdsource`	Password of the user or the absolute path of the encrypted password file. **Examples:** Windows: `set epmpwdsource="Example"` Linux/UNIX: `epmpwdsource="Example"`
`epmurlsource`	URL of the environment from which users are to be copied. **Examples:** Windows: `set epmurlsource="https://example.oraclecloud.com"` Linux/UNIX: `epmurlsource="https://example.oraclecloud.com"`
`epmidentitydomainsource`	Name of the identity domain used by the source environment. **Examples:** Windows: `set epmidentitydomainsource="example_source_dom"` Linux/UNIX: `epmidentitydomainsource="example_source_dom"`
`epmusertarget`	User name of a user with Identity Domain Administrator and Service Administrator roles in the target environment. **Examples:** Windows: `set epmusertarget="John.Doe"` Linux/UNIX: `set epmusertarget="John.Doe"`
`epmpwdtarget`	Password of the user or the absolute path of the encrypted password file. **Examples:** Windows: `set epmpwdtarget="Example1"` Linux/UNIX: `epmpwdtarget="Example1"`
`epmurltarget`	URL of the environment in which users are to be created. **Examples:** Windows: `set epmurltarget="https://example.oraclecloud.com"` Linux/UNIX: `epmurltarget="https://example.oraclecloud.com"`
`epmidentitydomaintarget`	Name of the identity domain used by the target environment. **Examples:** Windows: `set epmidentitydomaintarget="example_source_dom"` Linux/UNIX: `epmidentitydomaintarget="example_target_dom"`

Parameter	Description
proxyserverusername	The user name to authenticate a secure session with the proxy server that controls access to the internet. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverusername="Example"` Linux/UNIX: `proxyserverusername="Example"`
proxyserverpassword	The password to authenticate the user with the proxy server. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverpassword="examplePwd"` Linux/UNIX: `proxyserverpassword="examplePwd"`
proxyserverdomain	The name of the domain defined for the proxy server. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverdomain="exampleDom"` Linux/UNIX: `proxyserverdomain="exampleDom"`
emailtoaddress	Optionally, the email address to which the Role Assignment report is to be sent. The report is emailed only if this value is specified. **Example:** `emailtoaddress=john.doe@example.com`

Linux/UNIX

1. Create `replicateusers.sh` by copying the following script.

```
#!/bin/sh

userpassword="$1"

# USER DEFINED VARIABLES
#-----------------------
javahome="<JAVA HOME>"
epmautomatescript="<EPM AUTOMATE SCRIPT LOCATION>"
epmusersource="<EPM USER FOR SOURCE ENVIRONMENT>"
epmpwdsource="<EPM PASSWORD FOR SOURCE ENVIRONMENT>"
epmurlsource="<EPM URL FOR SOURCE ENVIRONMENT>"
epmidentitydomainsource="<EPM IDENTITY DOMAIN FOR SOURCE ENVIRONMENT>"
epmusertarget="<EPM USER FOR TARGET ENVIRONMENT>"
epmpwdtarget="<EPM PASSWORD FOR TARGET ENVIRONMENT>"
epmurltarget="<EPM URL FOR TARGET ENVIRONMENT>"
epmidentitydomaintarget="<EPM IDENTITY DOMAIN FOR TARGET ENVIRONMENT>"
proxyserverusername="<PROXY SERVER USER NAME>"
proxyserverpassword="<PROXY SERVER PASSWORD>"
proxyserverdomain="<PROXY SERVER DOMAIN>"
resetpassword="false"
emailtoaddress="<EMAIL TO ADDRESS>"
#-----------------------

roleassignmentreport="roleassignmentreport.csv"
usersreport="users.csv"
paramrequiredmessage='Syntax: replicateusers.sh "USER_PASSWORD"'

export JAVA_HOME=${javahome}
```

```
if [ "${userpassword}" == "" ]
then
    echo "USER_PASSWORD is missing. This is used to set the default
password for the replicated users."
    echo "${paramrequiredmessage}"
    exit
fi

echo "Replicate users script started"

# epmautomate login Source App as an IDM Admin
echo "Logging into source application at ${epmurlsource}"
${epmautomatescript} login ${epmusersource} ${epmpwdsource} $
{epmurlsource} ${epmidentitydomainsource} ${proxyserverusername} $
{proxyserverpassword} ${proxyserverdomain}
echo "Creating role assignment report: ${roleassignmentreport}"
${epmautomatescript} roleAssignmentReport ${roleassignmentreport}
if [[ "${emailtoaddress}" == *"@"* ]]
then
    ${epmautomatescript} sendMail $emailtoaddress "Role assignment report"
Body="Role assignment report is attached."
Attachments=$roleassignmentreport
fi
echo "Downloading role assignment report"
${epmautomatescript} downloadfile ${roleassignmentreport}
${epmautomatescript} deletefile ${roleassignmentreport}
${epmautomatescript} logout

awk -F, '{print $2","$3","$4","$1}' ${roleassignmentreport} | (read -r;
printf "%s\n" "$REPLY"; sort -u) > ${usersreport}

# epmautomate login Target App as an IDM Admin
echo "Logging into target application at ${epmurltarget}"
${epmautomatescript} login ${epmusertarget} ${epmpwdtarget} $
{epmurltarget} ${epmidentitydomaintarget} ${proxyserverusername} $
{proxyserverpassword} ${proxyserverdomain}
${epmautomatescript} deletefile ${usersreport} > /dev/null 2>&1
echo "Uploading file ${usersreport}"
${epmautomatescript} uploadfile ${usersreport}
echo "Adding users"
${epmautomatescript} addUsers ${usersreport} userPassword=${userpassword}
resetPassword=${resetpassword}
${epmautomatescript} deletefile ${usersreport}
${epmautomatescript} logout
rm deletefile*.log > /dev/null 2>&1

echo "Replicate users script completed"
```

2. Update `replicateusers.sh`. See the preceding table for information on the values you must specify. Additionally, you must specify the values for these properties:
   - `javahome`: the absolute path to the directory where Java is installed.
   - `epmautomatescript`: Location of `epmautomatescript.sh`; for example, `epmautomatescript="/home/user1/epmautomate/bin/epmautomate.sh"`

Replicating Predefined Role Assignments from One Environment to Another

Use the scripts in this section to clone predefined role assignments from one environment to another. The user running these scripts must have Service Administrator role in both environments.

> **Note:**
>
> **If you are using the PDF version of this document:** To avoid line breaks and footer information that will render these scripts unusable, copy them from the HTML version of this topic.

Windows

1. Create `replicatepredefineroles.ps1` by copying the following script.

```
# Replicate predefined roles script

param(
  [string]$epmusersource,
  [string]$epmpwdsource,
  [string]$epmurlsource,
  [string]$epmidentitydomainsource,
  [string]$epmusertarget,
  [string]$epmpwdtarget,
  [string]$epmurltarget,
  [string]$epmidentitydomaintarget,
  [string]$proxyserverusername,
  [string]$proxyserverpassword,
  [string]$proxyserverdomain,
  [string]$emailtoaddress
)

$roleassignmentreport="roleassignmentreport.csv"

function replicateroles
{
    # epmautomate login Source App as an IDM Admin
    echo "Logging into source application at ${epmurlsource}"
    epmautomate login ${epmusersource} ${epmpwdsource} ${epmurlsource} $
{epmidentitydomainsource} ${proxyserverusername} ${proxyserverpassword} $
{proxyserverdomain}
    echo "Creating role assignment report: ${roleassignmentreport}"
    epmautomate roleAssignmentReport ${roleassignmentreport}
    if (${emailtoaddress} -match "@") {
        epmautomate.bat sendMail $emailtoaddress "Role assignment report"
Body="Role assignment report is attached."
Attachments=$roleassignmentreport
    }
    echo "Downloading role assignment report"
    epmautomate downloadfile ${roleassignmentreport}
    epmautomate deletefile ${roleassignmentreport}
    epmautomate logout
```

```
    echo "Creating files to use with epmautomate assignRoles"

    Get-Content ${roleassignmentreport} | ForEach-Object {
        $user=$_.split(',')[0]
        $rolename=$_.split(',')[4]

        if ($rolename -like '*User' -And $rolename -notlike '*Power User')
{
            $rolenamearray=$rolename.split(" ")
            $arraysize=$rolenamearray.count
            $rolename="User"
            if ($arraysize.count -le 2) {
                echo "${user}" | Out-File -Append -Encoding "UTF8" "role-$
{rolename}.csv"
            }
        }
        elseif ($rolename -like '*Viewer') {
            $rolenamearray=$rolename.split(" ")
            $arraysize=$rolenamearray.count
            $rolename="Viewer"
            if ($arraysize -le 2) {
                echo "${user}" | Out-File -Append -Encoding "UTF8" "role-$
{rolename}.csv"
            }
        }
        elseif ($rolename -like '*Power User') {
            $rolenamearray=$rolename.split(" ")
            $arraysize=$rolenamearray.count
            $rolename="Power User"
            if ($arraysize -le 3) {
                echo "${user}" | Out-File -Append -Encoding "UTF8" "role-$
{rolename}.csv"
            }
        }
        elseif ($rolename -like '*Service Administrator') {
            $rolenamearray=$rolename.split(" ")
            $arraysize=$rolenamearray.count
            $rolename="Service Administrator"
            if ($arraysize -le 3) {
                echo "${user}" | Out-File -Append -Encoding "UTF8" "role-$
{rolename}.csv"
            }
        }
        elseif ($rolename -like 'Planner') {
            echo "${user}" | Out-File -Append -Encoding "UTF8" "role-
User.csv"
        }
    }

    # Add header and format
    $rolefiles = Get-ChildItem "role-*.csv"
    foreach ($rolefile in $rolefiles) {
        $rolefilecontent = Get-Content "$rolefile"
        $headerline='User Login'
        Set-Content $rolefile -value $headerline,$rolefilecontent
```

```
        $txt = [io.file]::ReadAllText("$rolefile") -replace "`r`n","`n"
        [io.file]::WriteAllText("$rolefile", $txt)
    }

    # epmautomate login Target App as an IDM Admin
    echo "Logging into target application at ${epmurltarget}"
    epmautomate login ${epmusertarget} ${epmpwdtarget} ${epmurltarget} $
{epmidentitydomaintarget} ${proxyserverusername} ${proxyserverpassword} $
{proxyserverdomain}

    $rolefiles = Get-ChildItem "role-*.csv"
    foreach ($rolefile in $rolefiles) {
        $rolenamecsv=$rolefile.BaseName.split('-')[1]
        $rolename=$rolenamecsv.split('.')[0]
        epmautomate deletefile "${rolefile}" | Out-Null
        echo "Uploading file ${rolefile}"
        epmautomate uploadfile "${rolefile}"
        echo "Assigning ${rolename} roles"
        epmautomate assignRole "role-${rolename}.csv" "${rolename}"
        epmautomate deletefile "role-${rolename}.csv"
    }
    epmautomate logout
    rm deletefile*.log | Out-Null
}

function init
{
    # delete ${role}.csv files
    $rolefiles = Get-ChildItem "role-*.csv"
    foreach ($rolefile in $rolefiles) {
        $rolefileexists=Test-Path $rolefile
        if ($rolefileexists) {
            rm "${rolefile}"
        }
    }
}

echo "Replicate predefined roles script started"
init
replicateroles
echo "Replicate predefined roles script completed"
```

2. Create `replicatepredefineroles.bat` by copying the following script.

```
@ECHO OFF
SET thisdir=%~dp0
SET scriptpath=%thisdir%replicatepredefinedroles.ps1

REM USER DEFINED VARIABLES
REM -----------------------
set epmusersource="<EPM USER FOR SOURCE ENVIRONMENT>"
set epmpwdsource="<EPM PASSWORD FOR SOURCE ENVIRONMENT>"
set epmurlsource="<EPM URL FOR SOURCE ENVIRONMENT>"
set epmidentitydomainsource="<EPM IDENTITY DOMAIN FOR SOURCE ENVIRONMENT>"
set epmusertarget="<EPM USER FOR TARGET ENVIRONMENT>"
set epmpwdtarget="<EPM PASSWORD FOR TARGET ENVIRONMENT>"
```

```
set epmurltarget="<EPM URL FOR TARGET ENVIRONMENT>"
set epmidentitydomaintarget="<EPM IDENTITY DOMAIN FOR TARGET ENVIRONMENT>"
set proxyserverusername="<PROXY SERVER USER NAME>"
set proxyserverpassword="<PROXY SERVER PASSWORD>"
set proxyserverdomain="<PROXY SERVER DOMAIN>"
set emailtoaddress="<EMAIL_TO_ADDRESS>"
REM -----------------------

PowerShell -NoProfile -ExecutionPolicy Bypass -Command "& '%scriptpath%' -
epmusersource '%epmusersource%' -epmpwdsource '%epmpwdsource%' -
epmurlsource '%epmurlsource%' -epmidentitydomainsource
'%epmidentitydomainsource%' -epmusertarget '%epmusertarget%' -epmpwdtarget
'%epmpwdtarget%' -epmurltarget '%epmurltarget%' -epmidentitydomaintarget
'%epmidentitydomaintarget%' -proxyserverusername '%proxyserverusername%' -
proxyserverpassword '%proxyserverpassword%' -proxyserverdomain
'%proxyserverdomain%' -emailtoaddress '%emailtoaddress%'"
```

3. Update `replicatepredefineroles.bat` as needed. See the following table for information on the values you must set for the properties in this file.

Updating `replicatepredefineroles.bat`

Parameter	Description
`epmusersource`	User name of a user with Identity Domain Administrator and Service Administrator roles in the source environment. **Examples:** Windows: `set epmusersource="jDoe"` Linux/UNIX: `epmusersource="jDoe"`
`epmpwdsource`	Password of the user or the absolute path of the encrypted password file. **Examples:** Windows: `set epmpwdsource="Example"` Linux/UNIX: `epmpwdsource="Example"`
`epmurlsource`	URL of the environment from which users are to be copied. **Examples:** Windows: `set epmurlsource="https://example.oraclecloud.com"` Linux/UNIX: `epmurlsource="https://example.oraclecloud.com"`
`epmidentitydomainsource`	Name of the identity domain used by the source environment. **Examples:** Windows: `set epmidentitydomainsource="example_source_dom"` Linux/UNIX: `epmidentitydomainsource="example_source_dom"`
`epmusertarget`	User name of a user with Identity Domain Administrator and Service Administrator roles in the target environment. **Examples:** Windows: `set epmusertarget="John.Doe"` Linux/UNIX: `set epmusertarget="John.Doe"`

Parameter	Description
epmpwdtarget	Password of the user or the absolute path of the encrypted password file. **Examples:** Windows: `set epmpwdtarget="Example1"` Linux/UNIX: `epmpwdtarget="Example1"`
epmurltarget	URL of the environment in which users are to be created. **Examples:** Windows: `set epmurltarget="https://example.oraclecloud.com"` Linux/UNIX: `epmurltarget="https://example.oraclecloud.com"`
epmidentitydomaintarget	Name of the identity domain used by the target environment. **Examples:** Windows: `set epmidentitydomaintarget="example_target_dom"` Linux/UNIX: `epmidentitydomaintarget="example_target_dom"`
proxyserverusername	The user name to authenticate a secure session with the proxy server that controls access to the internet. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverusername="Example"` Linux/UNIX: `proxyserverusername="Example"`
proxyserverpassword	The password to authenticate the user with the proxy server. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverpassword="examplePwd"` Linux/UNIX: `proxyserverpassword="examplePwd"`
proxyserverdomain	The name of the domain defined for the proxy server. Delete all occurrence of this property if not used. **Examples:** Windows: `set proxyserverdomain="exampleDom"` Linux/UNIX: `proxyserverdomain="exampleDom"`
emailtoaddress	Optionally, the email address to which the Role Assignment report is to be sent. The report is emailed only if this value is specified. **Example:** `emailtoaddress=john.doe@example.com`

Linux/UNIX

1. Create `replicatepredefineroles.sh` by copying the following script.

```
#!/bin/sh

# USER DEFINED VARIABLES
#-----------------------
javahome="<JAVA HOME>"
epmautomatescript="<EPM AUTOMATE SCRIPT LOCATION>"
epmusersource="<EPM USER FOR SOURCE ENVIRONMENT>"
epmpwdsource="<EPM PASSWORD FOR SOURCE ENVIRONMENT>"
```

```
epmurlsource="<EPM URL FOR SOURCE ENVIRONMENT>"
epmidentitydomainsource="<EPM IDENTITY DOMAIN FOR SOURCE ENVIRONMENT>"
epmusertarget="<EPM USER FOR TARGET ENVIRONMENT>"
epmpwdtarget="<EPM PASSWORD FOR TARGET ENVIRONMENT>"
epmurltarget="<EPM URL FOR TARGET ENVIRONMENT>"
epmidentitydomaintarget="<EPM IDENTITY DOMAIN FOR TARGET ENVIRONMENT>"
proxyserverusername="<PROXY SERVER USER NAME>"
proxyserverpassword="<PROXY SERVER PASSWORD>"
proxyserverdomain="<PROXY SERVER DOMAIN>"
emailtoaddress="<EMAIL TO ADDRESS>"
#-----------------------

roleassignmentreport="roleassignmentreport.csv"

export JAVA_HOME=${javahome}


replicateroles()
{
    # epmautomate login Source App as an DM Admin
    echo "Logging into source application at ${epmurlsource}"
    ${epmautomatescript} login ${epmusersource} ${epmpwdsource} $
{epmurlsource} ${epmidentitydomainsource} ${proxyserverusername} $
{proxyserverpassword} ${proxyserverdomain}
    echo "Creating role assignment report: ${roleassignmentreport}"
    ${epmautomatescript} roleAssignmentReport ${roleassignmentreport}
    if [[ "${emailtoaddress}" == *"@"* ]]
    then
        ${epmautomatescript} sendMail $emailtoaddress "Role assignment
report" Body="Role assignment report is attached."
Attachments=$roleassignmentreport
    fi
    echo "Downloading role assignment report"
    ${epmautomatescript} downloadfile ${roleassignmentreport}
    ${epmautomatescript} deletefile ${roleassignmentreport}
    ${epmautomatescript} logout

    echo "Creating files to use with epmautomate assignRoles"
    while read line
    do
        user=$(echo "${line}" | cut -d',' -f1)
        rolename=$(echo "${line}" | cut -d',' -f5)

        if [[ "$rolename" == *"User" ]] && [[ "$rolename" != "*Power
User" ]]
        then
            count=$(echo "${rolename}" | wc -w);
            rolename="User"
            if [[ $count -le 2 ]]
            then
                echo "${user}" >> "role-${rolename}.csv"
            fi
        elif [[ "$rolename" == *"Viewer" ]]
        then
            count=$(echo "${rolename}" | wc -w);
            rolename="Viewer"
```

```
            if [[ $count -le 2 ]]
            then
                echo "${user}" >> "role-${rolename}.csv"
            fi
        elif [[ "$rolename" == *"Power User" ]]
        then
            count=$(echo "${rolename}" | wc -w);
            rolename="Power User"
            if [[ $count -le 3 ]]
            then
                echo "${user}" >> "role-${rolename}.csv"
            fi
        elif [[ "$rolename" == *"Service Administrator" ]]
        then
            count=$(echo "${rolename}" | wc -w);
            rolename="Service Administrator"
            if [[ $count -le 3 ]]
            then
                echo "${user}" >> "role-${rolename}.csv"
            fi
        elif [[ "$rolename" == "Planner" ]]
        then
                echo "${user}" >> "role-User.csv"
        fi
    done < ${roleassignmentreport}

    # write header line
    for f in role-*.csv
    do
        sed -i '1iUser Login' "$f"
    done

    # epmautomate login Target App as an IDM Admin
    echo "Logging into target application at ${epmurltarget}"
    ${epmautomatescript} login ${epmusertarget} ${epmpwdtarget} $
{epmurltarget} ${epmidentitydomaintarget} ${proxyserverusername} $
{proxyserverpassword} ${proxyserverdomain}

    for rolefile in role-*.csv
    do
        rolenamecsv=$(echo "$rolefile" | cut -d'-' -f2)
        rolename=$(echo "$rolenamecsv" | cut -d'.' -f1)
        ${epmautomatescript} deletefile "${rolefile}" > /dev/null 2>&1
        echo "Uploading file ${rolefile}"
        ${epmautomatescript} uploadfile "${rolefile}"
        echo "Assigning roles"
        ${epmautomatescript} assignrole "${rolefile}" "${rolename}"
        ${epmautomatescript} deletefile "${rolefile}"
    done

    ${epmautomatescript} logout
    rm deletefile*.log > /dev/null 2>&1
}

init()
{
```

```
    # delete role-${role}.csv files
    for f in role-*.csv
    do
        rm "$f" > /dev/null 2>&1
    done
}

echo "Replicate predefined roles script started"
init
replicateroles
echo "Replicate predefined roles script completed"
```

2. Update `replicatepredefineroles.sh`. See the preceding table for information on the values you must specify. Additionally, you must specify the values for these properties:
   - `javahome`: the absolute path to the directory where Java is installed.
   - `epmautomatescript`: Location of `epmautomatescript.sh`; for example, `epmautomatescript="/home/user1/epmautomate/bin/epmautomate.sh"`

Create a Quarterly EPM Cloud Upgrade Cadence

Use these scripts to create a self-service solution to skip updates so that Oracle Enterprise Performance Management Cloud environments are updated on a quarterly basis with a two-week test cycle. In this case, the production environments are updated two weeks after test environments.

This script may also be used to skip every other monthly update, if needed. By default, EPM Cloud applies a monthly update to your environments. You use the skipUpdate command to skip the applying of monthly updates to an environment or to view current skip update requests. You can automate the manual running of the `skipUpdate` commands by using the scripts included in this section. These scripts automate the skip update process so that the updates are applied quarterly or every other month.

> **Note:**
>
> 1. You cannot skip updates for more than two consecutive months. For example, the script throws an error if you try to have an EPM Cloud environment updated only in February, June, and November.
> 2. All updates that have happened during the intervening period are applied to your environment during the next update. For example, assume that you use this script to schedule quarterly updates to occur only in February, May, August, and November. In this case, the May update, for example, will apply all applicable EPM Cloud monthly updates and patches that were released after the February update to your environment. The maintenance process may take more time than usual when the update is applied.
> 3. This script sets up the update cadence for one quarter only. Run this script on a monthly basis to ensure that the update cadence is configured for the whole year.

- Windows Script and Instructions
- UNIX/Linux Script and Instructons
- Groovy Script

Running the Script

1. To run the Windows and Linux/UNIX scripts:
   a. Create the `input.properties` file and update it with information for your environment. Save the file in a local directory. Contents of this file differs depending on your operating system.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
   b. Create `skip_update.ps1` (Windows PowerShell) or `skip_update.sh` (Linux/UNIX) bash script and save it in the directory where `input.properties` is located.
   c. Launch the script.
      - Linux/UNIX: run `./skip_update.sh`.
      - Windows PowerShell: run `skip_update.ps1`.
2. To run the Groovy script, use the Groovy screen in an EPM Cloud business process or automate the script execution using runBusinessRule. For information on running Groovy Script using EPM Automate, see Running Commands without Installing EPM Automate.

Windows Script and Instructions

Create `input.properties` and `skip_update.ps1` by copying the scripts in this section.

1. Create `input.properties` by copying the following script:

```
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updatemonths=02,05,08,11
```

2. Update `input.properties` by specifying parameter values.

Table 3-11 input.properties Parameters

Parameter	Description
`username`	User name of a Service Administrator.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`url`	URL of the environment on which you want to set the non-monthly update cadence.
`updatemonths`	A comma separated list of months when Oracle Enterprise Performance Management Cloud updates should be applied to the environment identified by the `url` parameter. For example, `updatemonths=02,05,08,11`. Months must be specified as two digits: 01 for January through 12 for December. Be sure to include a preceding zero for January through September. The script attempts to run the skipUpdatecommand for the months not included in the `updatemonths` parameter value. For example, if you specify `updatemonths=02,05,08,11`, the script tries to set skip update flags for January, March, April, June, July, September, October, and December so that updates are made only in February, May, August, and November.

3. Create `skip_updates.ps1` by copying the following script:

```
# Skip Update PowerShell script

$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -
raw)
$username="$($inputproperties.username)"
$password="$($inputproperties.password)"
$url="$($inputproperties.url)"
$updatemonths="$($inputproperties.updatemonths)"

$monthsarr = ("01","02","03","04","05","06","07","08","09","10","11","12")
$global:monthsarrfromcurrent = @()
$global:yearsarrfromcurrent = @()
$updatemonthsarr = $updatemonths.Split(",")
$currentyear=Get-Date -Format yy
$currentmonth=Get-Date -Format MM
$nextyear=[int]$currentyear+1

function populateFromCurrentArrays() {
    $startposition = 0

    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${currentmonth} -eq $monthsarr[$i]) {
            $startposition=$i
        break
    }
    }

    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${i} -ge ${startposition}) {
            $global:monthsarrfromcurrent += $monthsarr[$i]
            $global:yearsarrfromcurrent += $currentyear
        }
    }

    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${i} -lt ${startposition}) {
            $global:monthsarrfromcurrent += $monthsarr[$i]
            $global:yearsarrfromcurrent += $nextyear
        }
    }
}

function skipUpdateAdd($yearnumber, $monthnumber) {
    echo "Running: epmautomate.bat skipUpdate add version=${yearnumber}.$
{monthnumber} comment=`"adding skipUpdate`""
    epmautomate skipUpdate add version=${yearnumber}.${monthnumber}
comment="adding skipUpdate"
}

function processSkipUpdates() {
    $addcount = 0

    echo "Running: epmautomate.bat login ${username} ${password} ${url}"
    epmautomate login ${username} ${password} ${url}
```

```
    echo "Running: epmautomate.bat skipUpdate remove"
    epmautomate skipUpdate remove

    for ($i = 0; $i -le ($global:monthsarrfromcurrent.length - 1); $i++) {
        $match = 1

        if (${addcount} -eq 2) {
            echo "Two skip update add calls have been made. No more will
be attempted."
            break
    }

        for ($j = 0; $j -le ($updatemonthsarr.length - 1); $j++) {
        if ($global:monthsarrfromcurrent[$i] -eq $updatemonthsarr[$j]) {
                $match = 0
        break
        }
    }

        if (${match} -eq 1) {

skipUpdateAdd $global:yearsarrfromcurrent[$i] $global:monthsarrfromcurrent[
$i]
            $addcount += 1
    }
    }

    echo "Running: epmautomate.bat skipUpdate list"
    epmautomate skipUpdate list
    echo "Running: epmautomate.bat logout"
    epmautomate logout
}

function compareUpdateMonths($thismonth, $nextmonth) {
    $nextmonthorig=${nextmonth}

    if (${nextmonth} -lt ${thismonth}) {
        $nextmonth+=12
    }

    $monthdiff = $nextmonth - $thismonth

    if (${monthdiff} -gt 3) {
        echo "There are more than 2 months skipped from month ${thismonth}
to month ${nextmonthorig}. Please correct updatemonths in input.properties
so that there are not more than two months skipped between each update
month. Exiting."
        exit 1
    }
}

function validateUpdateMonths() {
    for ($i = 0; $i -le ($updatemonthsarr.length - 1); $i++) {
        $nextint = $i + 1
    $thisupdatemonth = $updatemonthsarr[$i]
        $thisupdatemonthint=[int]$thisupdatemonth
```

```
    $nextupdatemonth=$updatemonthsarr[$nextint]
        $nextupdatemonthint=[int]$nextupdatemonth

        if (${nextupdatemonth} -eq "") {
            $nextupdatemonth=$updatemonthsarr[0]
            $nextupdatemonthint=[int]$nextupdatemonth
    }

        compareUpdateMonths $thisupdatemonthint $nextupdatemonthint
    }
}

validateUpdateMonths
populateFromCurrentArrays
processSkipUpdates
```

UNIX/Linux Script and Instructons

Create `input.properties` and `skip_update.sh` by copying the scripts in this section.

1. Create `input.properties` by copying the following script:

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updatemonths=02,05,08,11
```

2. Update `input.properties` by specifying parameter values.

Table 3-12 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`).
`username`	User name of a Service Administrator.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`url`	URL of the environment on which you want to set the non-monthly update cadence.

Table 3-12 (Cont.) input.properties Parameters

Parameter	Description
updatemonths	A comma separated list of months when Oracle Enterprise Performance Management Cloud updates should be applied to the environment identified by the url parameter. For example, updatemonths=02,05,08,11. Months must be specified as two digits; include a preceding zero for January through September. The script attempts to run the skipUpdatecommand for the months not included in the updatemonths parameter value. For example, if you specify updatemonths=02,05,08,11, the script tries to set skip update flags for January, March, April, June, July, September, October, and December so that updates are made only in February, May, August, and November.

3. Create skip_updates.sh by copying the following script:

```
#!/bin/sh

. ./input.properties
export JAVA_HOME=${javahome}

declare -a monthsarr=(01 02 03 04 05 06 07 08 09 10 11 12)
declare -a monthsarrfromcurrent
declare -a yearsarrfromcurrent
updatemonthsarr=( $(echo "${updatemonths}" | sed 's/,/ /g') )
currentyear=$(date +%y)
nextyear=$((currentyear+1))
currentmonth=$(date +%m)

populateFromCurrentArrays() {
    for i in ${!monthsarr[@]}
    do
        if [[ "${currentmonth}" == "${monthsarr[$i]}" ]]
        then
            startposition=$i
            break
        fi
    done

    for i in ${!monthsarr[@]}
    do
        if [[ ${i} -ge ${startposition} ]]
        then
            monthsarrfromcurrent=("${monthsarrfromcurrent[@]}" "$
{monthsarr[$i]}")
            yearsarrfromcurrent=("${yearsarrfromcurrent[@]}" "$
{currentyear}")
        fi
    done

    for i in ${!monthsarr[@]}
    do
        if [[ ${i} -lt ${startposition} ]]
        then
```

```
            monthsarrfromcurrent=("${monthsarrfromcurrent[@]}" "$
{monthsarr[$i]}")
            yearsarrfromcurrent=("${yearsarrfromcurrent[@]}" "${nextyear}")
        fi
    done
}

skipUpdateAdd() {
    local yearnumber="$1"
    local monthnumber="$2"

    echo "Running: ${epmautomatescript} skipUpdate add version=$
{yearnumber}.${monthnumber} comment=\"adding skipUpdate\""
    ${epmautomatescript} skipUpdate add version=${yearnumber}.$
{monthnumber} comment="adding skipUpdate"
}

processSkipUpdates() {
    local addcount=0

    echo "Running: ${epmautomatescript} login ${username} ${password} $
{url}"
    ${epmautomatescript} login ${username} ${password} ${url}
    echo "Running: ${epmautomatescript} skipUpdate remove"
    ${epmautomatescript} skipUpdate remove

    for i in ${!monthsarrfromcurrent[@]}
    do
        local match=1

        if [[ ${addcount} -eq 2 ]]
        then
            echo "Two skip update add calls have been made. No more will
be attempted."
            break
        fi

        for j in ${!updatemonthsarr[@]}
        do
            if [[ "${monthsarrfromcurrent[$i]}" == "$
{updatemonthsarr[$j]}" ]]
            then
                match=0
                break
            fi
        done

        if [[ ${match} -eq 1 ]]
        then
            skipUpdateAdd ${yearsarrfromcurrent[$i]} "$
{monthsarrfromcurrent[$i]}"
            addcount=$((addcount+1))
        fi
    done

    echo "Running: ${epmautomatescript} skipUpdate list"
```

```
    ${epmautomatescript} skipUpdate list
    echo "Running: ${epmautomatescript} logout"
    ${epmautomatescript} logout
}

compareUpdateMonths() {
    local thismonth=$1
    local nextmonth=$2
    local nextmonthorig=${nextmonth}

    if [[ ${nextmonth} -lt ${thismonth} ]]
    then
        nextmonth=$((nextmonth+12))
    fi

    monthdiff=$((nextmonth-thismonth))

    if [[ ${monthdiff} -gt 3 ]]
    then
        echo "There are more than 2 months skipped from month ${thismonth}
to month ${nextmonthorig}. Please correct updatemonths in input.properties
so that there are not more than two months skipped between each update
month. Exiting."
        exit 1
    fi
}

validateUpdateMonths() {
    for i in ${!updatemonthsarr[@]}
    do
        nextint=$((i+1))
        thisupdatemonth="${updatemonthsarr[$i]}"
        thisupdatemonthint=${thisupdatemonth#0}
        nextupdatemonth="${updatemonthsarr[$nextint]}"
        nextupdatemonthint=${nextupdatemonth#0}

        if [[ ${nextupdatemonth} == "" ]]
        then
            nextupdatemonth="${updatemonthsarr[0]}"
            nextupdatemonthint=${nextupdatemonth#0}
        fi

        compareUpdateMonths ${thisupdatemonthint} ${nextupdatemonthint}
    done
}

validateUpdateMonths
populateFromCurrentArrays
processSkipUpdates
```

Groovy Script

If passwords contain special characters, see Handling Special Characters. Also, be sure to replace these parameter values to suit your environments:

Table 3-13 Parameters to Change

Parameter	Description
user	User name of a Service Administrator.
password	Password of the Service Administrator or the name and location of the encrypted password file.
url	URL of the environment on which you want to set the non-monthly update cadence.
updatemonths	A comma separated list of months when Oracle Enterprise Performance Management Cloud updates should be applied to the environment identified by the `url` parameter. For example, `updatemonths=02,05,08,11`. Months must be specified as two digits: 01 for January through 12 for December. Be sure to include a preceding zero for January through September. The script attempts to run the skipUpdatecommand for the months not included in the `updatemonths` parameter value. For example, if you specify `updatemonths=02,05,08,11`, the script tries to set skip update flags for January, March, April, June, July, September, October, and December so that updates are made only in February, May, August, and November.

```
import java.text.SimpleDateFormat

String user = 'service_administrator'
String password = 'examplePWD'
String url = 'example_EPM_URL'
String updatemonths = '02,05,08,11'

def currentdate = new Date()
def yf = new SimpleDateFormat("yy")
def mf = new SimpleDateFormat("MM")
String[] monthsarr = ["01", "02", "03", "04", "05", "06", "07", "08", "09",
"10", "11", "12"]
List<String> monthsarrfromcurrent = new ArrayList<>()
List<String> yearsarrfromcurrent = new ArrayList<>()
String currentyear = yf.format(currentdate)
String nextyear = (currentyear.toInteger() + 1).toString()
String currentmonth = mf.format(currentdate)

String[] updateMonthsStringArr = updatemonths.split(',');
def updatemonthsarr = new int[updateMonthsStringArr.length];
for(int i = 0; i < updateMonthsStringArr.length; i++)
{
    updatemonthsarr[i] = Integer.parseInt(updateMonthsStringArr[i]);
}

def LogMessage(String message) {
    def date = new Date()
    def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss")
    println('[' + sdf.format(date) + '][GROOVY] ' + message);
}

def LogOperationStatus(EpmAutomateStatus opstatus) {
```

```
    def returncode = opstatus.getStatus()
    if (returncode != 0){
        LogMessage(opstatus.getOutput())
    }
    LogMessage('return code: ' + returncode)
}

int CompareUpdateMonths(int thismonth, int nextmonth) {
    int nextmonthorig = nextmonth

    if (nextmonth < thismonth) {
        nextmonth = nextmonth + 12
    }

    int monthdiff = nextmonth - thismonth

    if (monthdiff > 3) {
        LogMessage('There are more than 2 months skipped from month ' +
thismonth + ' to month ' + nextmonthorig + '. Please correct updatemonths so
that there are not more than two months skipped between each update month.
Exiting.')
        return 1
    }

    return 0
}

int ValidateUpdateMonths(int[] updatemonthsarr) {
    for(int i = 0; i < updatemonthsarr.length; i++)
    {
        int nextint = i + 1
        String nextupdatemonth = ""
        int nextupdatemonthint = 0
        String thisupdatemonth = updatemonthsarr[i]
        int thisupdatemonthint = thisupdatemonth.toInteger()

        if (nextint < updatemonthsarr.length) {
            nextupdatemonth = updatemonthsarr[nextint]
        } else {
            nextupdatemonth = updatemonthsarr[0]
        }

        nextupdatemonthint = nextupdatemonth.toInteger()

        int returncode = CompareUpdateMonths(thisupdatemonthint,
nextupdatemonthint)
        if (returncode > 0) {
            return 1
        }
    }
    return 0
}

def SkipUpdateAdd(EpmAutomate automate, String yearnumber, String
monthnumber) {
    String yeardotmonth = yearnumber + '.' + monthnumber
```

```
    LogMessage('Running: epmautomate skipUpdate add version=' + yeardotmonth
+ ' comment=\"adding skipUpdate\"')
    EpmAutomateStatus status = automate.execute('skipupdate','add','version='
+ yeardotmonth,'comment=\"adding skipUpdate\"')
    LogOperationStatus(status)
}

LogMessage('Starting skip update processing')
EpmAutomate automate = getEpmAutomate()

// validate update months
int returncode = ValidateUpdateMonths(updatemonthsarr)
if (returncode != 0) {
    return 1
}

// populate arrays
int startposition = 0
for(int i = 0; i < monthsarr.length; i++)
{
    if (currentmonth == monthsarr[i]) {
        startposition = i
        break
    }
}

for(int i = 0; i < monthsarr.length; i++)
{
    if (i >= startposition) {
        monthsarrfromcurrent.add(monthsarr[i])
        yearsarrfromcurrent.add(currentyear)
    }
}

for(int i = 0; i < monthsarr.length; i++)
{
    if (i <= startposition) {
        monthsarrfromcurrent.add(monthsarr[i])
        yearsarrfromcurrent.add(nextyear)
    }
}

// process skip updates
LogMessage("Operation: encrypt " + password + " oracleKey password.epw")
EpmAutomateStatus status =
automate.execute('encrypt',password,"oracleKey","password.epw")
LogOperationStatus(status)

LogMessage("Operation: login " + user + " password.epw " + url)
status = automate.execute('login',user,"password.epw",url)
LogOperationStatus(status)

LogMessage('Running: epmautomate skipUpdate remove')
status = automate.execute('skipupdate','remove')
LogOperationStatus(status)
```

```
int addcount = 0

for (int i = 0; i < monthsarrfromcurrent.size(); i++) {
    int match = 1

    if (addcount == 2){
        LogMessage('Two skip update add calls have been made. No more will be
attempted.')
        break
    }

    for(int j = 0; j < updatemonthsarr.length; j++) {

        if (Integer.parseInt(monthsarrfromcurrent.get(i)) ==
updatemonthsarr[j]) {
            match = 0
            break
        }
    }

    if (match == 1) {
        SkipUpdateAdd(automate, yearsarrfromcurrent.get(i),
monthsarrfromcurrent.get(i))
        addcount+=1
    }
}

LogMessage('Running: epmautomate skipUpdate list')
status = automate.execute('skipupdate','list')
LogOperationStatus(status)

LogMessage('Running: epmautomate logout')
status = automate.execute('logout')
LogOperationStatus(status)

LogMessage('Skip update processing completed')
```

Create a Quarterly EPM Cloud Upgrade Cadence with Six Week Test Cycles

Use the script in this section to create a self-service solution to skip updates so that Oracle Enterprise Performance Management Cloud environments are updated on a quarterly basis with a six-week test cycle. In this case, the production environments are updated six weeks after the test environments.

By default, EPM Cloud applies a monthly update to your environments. You use the skipUpdate command to skip the applying of monthly updates to an environment or to view current skip update requests. You can automate the manual running of the `skipUpdate` commands by using the scripts included in this section. These scripts automate the skip update process so that the updates are applied on a quarterly basis with six weeks test cycle.

Note:

1. You cannot skip updates for more than three consecutive months. This script throws an error if you try to have an EPM Cloud environment updated only in February and October.
2. All updates that have happened during the intervening period are applied to your environment during the next update. For example, assume that you use this script to schedule quarterly updates to occur only for February, May, August, and November updates. In this case, the May update, for example, will apply all applicable EPM Cloud monthly updates and patches that were released after the February update to your environment. The maintenance process may take more time than usual when the update is applied.
3. This script sets up the update cadence for one quarter only.
   Sample Scenario: The test environment update cycle is established as the first Fridays of February (24.02 update), May (24.05 update), August (24.08 update), and November (24.11 update). The Production Environment will be updated on the third Fridays of March (24.02 update) with the version that was used to update the test environment on the first Friday of February (24.02 update). Similar update to the production environment will occur on the third week of June (24.05 update), September (24.08 update), and December (24.11 update). In this scenario, the production environment is not updated to the current update, but with the update that is currently on the test environment.

Windows Sample Script

Create `skip_update.ps1` by copying the following script. Store it in a local directory. See Running the Script for information on running this script:

```
# Skip Update PowerShell script

$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$password="$($inputproperties.password)"
$url="$($inputproperties.url)"
$updateversions="$($inputproperties.updateversions)"
$podtype="$($inputproperties.podtype)"
$proxyserverusername="$($inputproperties.proxyserverusername)"
$proxyserverpassword="$($inputproperties.proxyserverpassword)"
$proxyserverdomain="$($inputproperties.proxyserverdomain)"

echo "Starting skip_update.ps1 script."

$monthsarr = ("01","02","03","04","05","06","07","08","09","10","11","12")
$global:monthsarrfromcurrent = @()
$global:yearsarrfromcurrent = @()
$updateversionsarr = $updateversions.Split(",")
$currentyear=Get-Date -Format yy
$currentmonth=Get-Date -Format MM
$nextyear=[int]$currentyear+1

function populateFromCurrentArrays() {
    $startposition = 0
```

```
    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${currentmonth} -eq $monthsarr[$i]) {
            if (${podtype} -eq "prod") {
                if (${updateversionsarr} -contains ${currentmonth}) {
                    $startposition=$i-2
        } else {
                    $startposition=$i-1
                }
                } else {
                if (${updateversionsarr} -contains ${currentmonth}) {
                    $startposition=$i
        } else {
                    $startposition=$i-1
        }
                }
            break
    }
    }

    if (${startposition} -lt 0) {
        $startposition=$startposition+12
    }

    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${i} -ge ${startposition}) {
            $global:monthsarrfromcurrent += $monthsarr[$i]
            $global:yearsarrfromcurrent += $currentyear
        }
    }

    for ($i = 0; $i -le ($monthsarr.length - 1); $i++) {
    if (${i} -lt ${startposition}) {
            $global:monthsarrfromcurrent += $monthsarr[$i]
            $global:yearsarrfromcurrent += $nextyear
        }
    }
}

function skipUpdateAdd($yearnumber, $monthnumber) {
    echo "Running: epmautomate.bat skipUpdate add version=${yearnumber}.$
{monthnumber} comment=`"adding skipUpdate`""
    epmautomate skipUpdate add version=${yearnumber}.${monthnumber}
comment="adding skipUpdate"
}

function processSkipUpdates() {
    $addcount = 0
    $countlimit = 0

    if (${podtype} -eq "prod") {
        $countlimit = 3
    } else {
        $countlimit = 2
    }
```

```
    if ((${proxyserverusername} -eq "") -And (${proxyserverpassword} -eq "") -
And (${proxyserverdomain} -eq "")) {
        echo "Running: epmautomate.bat login ${username} ${password} ${url}"
        epmautomate login ${username} ${password} ${url}
    } else {
        echo "Running: epmautomate.bat login ${username} ${password} ${url}
ProxyServerUserName=${proxyserverusername} ProxyServerPassword=$
{proxyserverpassword} ProxyServerDomain=${proxyserverdomain}"
        epmautomate login ${username} ${password} ${url} ProxyServerUserName=$
{proxyserverusername} ProxyServerPassword=${proxyserverpassword}
ProxyServerDomain=${proxyserverdomain}
    }

    echo "Running: epmautomate.bat skipUpdate remove"
    epmautomate skipUpdate remove

    for ($i = 0; $i -le ($global:monthsarrfromcurrent.length - 1); $i++) {
        $match = 1

        if (${addcount} -eq ${countlimit}) {
            echo "Update calls are completed. No more will be attempted."
            break
    }

        for ($j = 0; $j -le ($updateversionsarr.length - 1); $j++) {
        if ((${currentmonth} -eq $updateversionsarr[$j]) -And (${addcount} -
gt 0)) {
                $match = 1
        break
        }

        if (($global:monthsarrfromcurrent[$i] -eq $updateversionsarr[$j]) -
And (${addcount} -eq 0)){
                $match = 0
        break
        }
    }

        if (${match} -eq 1) {

skipUpdateAdd $global:yearsarrfromcurrent[$i] $global:monthsarrfromcurrent[$i]
            $addcount += 1
    }
    }

    echo "Running: epmautomate.bat skipUpdate list"
    epmautomate skipUpdate list
    echo "Running: epmautomate.bat logout"
    epmautomate logout
}

function compareUpdateMonths($thismonth, $nextmonth) {
    $nextmonthorig=${nextmonth}

    if (${nextmonth} -lt ${thismonth}) {
        $nextmonth+=12
```

```
    }

    $monthdiff = $nextmonth - $thismonth

    if (${monthdiff} -gt 4) {
        echo "There are more than 3 versions skipped from version $
{thismonth} to version ${nextmonthorig}. Please correct updateversions in
input.properties so that there are not more than three versions skipped
between each update version. Exiting."
        exit 1
    }
}

function validateUpdateVersions() {
    for ($i = 0; $i -le ($updateversionsarr.length - 1); $i++) {
        $nextint = $i + 1
    $thisupdatemonth = $updateversionsarr[$i]
        $thisupdatemonthint=[int]$thisupdatemonth
    $nextupdatemonth=$updateversionsarr[$nextint]
        $nextupdatemonthint=[int]$nextupdatemonth

        if (${nextupdatemonth} -eq "") {
            $nextupdatemonth=$updateversionsarr[0]
            $nextupdatemonthint=[int]$nextupdatemonth
    }

        compareUpdateMonths $thisupdatemonthint $nextupdatemonthint
    }
}

validateUpdateVersions
populateFromCurrentArrays
processSkipUpdates
```

Linux/UNIX Sample Script

Create `skip_update.sh` by copying the following script. Store it in a local directory. See Running the Script for information on running this script:

```
#!/bin/sh

. ./input.properties

echo "Starting skip_update.sh script."

export JAVA_HOME=${javahome}

declare -a monthsarr=(01 02 03 04 05 06 07 08 09 10 11 12)
declare -a monthsarrfromcurrent
declare -a yearsarrfromcurrent
updateversionsarr=( $(echo "${updateversions}" | sed 's/,/ /g') )
currentyear=$(date +%y)
nextyear=$((currentyear+1))
currentmonth=$(date +%m)

populateFromCurrentArrays() {
```

```
    local startposition=0

    for i in ${!monthsarr[@]}
    do
        if [[ "${currentmonth}" == "${monthsarr[$i]}" ]]
        then
            if [[ "${podtype}" == "prod" ]]
            then
                if [[ ${updateversionsarr[@]} =~ ${currentmonth} ]]
                then
                    startposition=$((i-2))
                else
                    startposition=$((i-1))
                fi
                break
            else
                if [[ ${updateversionsarr[@]} =~ ${currentmonth} ]]
                then
                    startposition=$i
                else
                    startposition=$((i-1))
                fi
                break
            fi
        fi
    done

    if [[ ${startposition} -lt 0 ]]
    then
        startposition=$((startposition+12))
    fi

    for i in ${!monthsarr[@]}
    do
        if [[ ${i} -ge ${startposition} ]]
        then
            monthsarrfromcurrent=("${monthsarrfromcurrent[@]}" "$
{monthsarr[$i]}")
            yearsarrfromcurrent=("${yearsarrfromcurrent[@]}" "${currentyear}")
        fi
    done

    for i in ${!monthsarr[@]}
    do
        if [[ ${i} -lt ${startposition} ]]
        then
            monthsarrfromcurrent=("${monthsarrfromcurrent[@]}" "$
{monthsarr[$i]}")
            yearsarrfromcurrent=("${yearsarrfromcurrent[@]}" "${nextyear}")
        fi
    done
}

skipUpdateAdd() {
    local yearnumber="$1"
    local monthnumber="$2"
```

```
    echo "Running: ${epmautomatescript} skipUpdate add version=${yearnumber}.${monthnumber} comment=\"adding skipUpdate\""
    ${epmautomatescript} skipUpdate add version=${yearnumber}.${monthnumber} comment="adding skipUpdate"
}

processSkipUpdates() {
    local addcount=0
    local countlimit=0

    if [[ "${podtype}" == "prod" ]]
    then
        countlimit=3
    else
        countlimit=2
    fi

    if [[ "${proxyserverusername}" == "" ]] && [[ "${proxyserverpassword}" == "" ]] && [[ "${proxyserverdomain}" == "" ]]
    then
        echo "Running: ${epmautomatescript} login ${username} ${password} ${url}"
        ${epmautomatescript} login ${username} ${password} ${url}
    else
        echo "Running: ${epmautomatescript} login ${username} ${password} ${url} ProxyServerUserName=${proxyserverusername} ProxyServerPassword=${proxyserverpassword} ProxyServerDomain=${proxyserverdomain}"
        ${epmautomatescript} login ${username} ${password} ${url} ProxyServerUserName=${proxyserverusername} ProxyServerPassword=${proxyserverpassword} ProxyServerDomain=${proxyserverdomain}
    fi
    echo "Running: ${epmautomatescript} skipUpdate remove"
    ${epmautomatescript} skipUpdate remove

    for i in ${!monthsarrfromcurrent[@]}
    do
        local match=1

        if [[ ${addcount} -eq ${countlimit} ]]
        then
            echo "Update add calls are completed. No more will be attempted."
            break
        fi

        for j in ${!updateversionsarr[@]}
        do
            if [[ "${currentmonth}" == "${updateversionsarr[$j]}" ]] && [[ ${addcount} -gt 0 ]]
            then
                match=1
                break
            fi

            if [[ "${monthsarrfromcurrent[$i]}" == "${updateversionsarr[$j]}" ]] && [[ ${addcount} -eq 0 ]]
```

```
            then
                match=0
                break
            fi
        done

        if [[ ${match} -eq 1 ]]
        then
            skipUpdateAdd ${yearsarrfromcurrent[$i]} "$
{monthsarrfromcurrent[$i]}"
            addcount=$((addcount+1))
        fi
    done

    echo "Running: ${epmautomatescript} skipUpdate list"
    ${epmautomatescript} skipUpdate list
    echo "Running: ${epmautomatescript} logout"
    ${epmautomatescript} logout
}

compareUpdateMonths() {
    local thismonth=$1
    local nextmonth=$2
    local nextmonthorig=${nextmonth}

    if [[ ${nextmonth} -lt ${thismonth} ]]
    then
        nextmonth=$((nextmonth+12))
    fi

    monthdiff=$((nextmonth-thismonth))

    if [[ ${monthdiff} -gt 4 ]]
    then
        echo "There are more than 3 versions skipped from version $
{thismonth} to version ${nextmonthorig}. Please correct updateversions in
input.properties so that there are not more than three versions skipped
between each update version. Exiting."
        exit 1
    fi
}

validateUpdateVersions() {
    for i in ${!updateversionsarr[@]}
    do
        nextint=$((i+1))
        thisupdatemonth="${updateversionsarr[$i]}"
        thisupdatemonthint=${thisupdatemonth#0}
        nextupdatemonth="${updateversionsarr[$nextint]}"
        nextupdatemonthint=${nextupdatemonth#0}

        if [[ ${nextupdatemonth} == "" ]]
        then
            nextupdatemonth="${updateversionsarr[0]}"
            nextupdatemonthint=${nextupdatemonth#0}
        fi
```

```
        compareUpdateMonths ${thisupdatemonthint} ${nextupdatemonthint}
    done
}

validateUpdateVersions
populateFromCurrentArrays
processSkipUpdates
```

Server-Side Groovy Script

Create `skip_update.groovy` Groovy script by copying the following script and then updating it. See Running the Script for information on running this script:
Update the following variables in this groovy script:

- `username` User name of a Service Administrator on the environment on which you want to set the non-monthly update cadence.
- `password` Password of the Service Administrator or the name and location of the encrypted password file.
- `url` URL of the environment on which you want to set the non-monthly update cadence.
- `updateversions` A comma separated list of EPM Cloud updates that should be applied to the environment identified by the `url` parameter. For example, `updateversions=02,05,08,11`.
  Versions must be specified as two digits; include a preceding zero for updates 01 through 09. The script attempts to run the skipUpdate command for the updates not included in the `updateversions` parameter value. For example, if you specify `updateversions=02,05,08,11`, the script tries to set skip update flags for the 01 (January), 03 (March), 04 (April), 06 (June), 07 (July), 09 (September), 10 (October), and 12 (December) updates. In this case, EPM Cloud updates 02 (February), 05 (May), 08 (August), and 11 (November) are applied to the environment.
- `podtype` EPM Cloud environment type. Valid values are `test` and `prod`.
- `proxyserverusername` The user name to authenticate a secure session with the proxy server that controls access to the internet.
- `proxyserverpassword` The password to authenticate the user with the proxy server.
- `proxyserverdomain` The name of the domain defined for the proxy server.

> **Note:**
>
> If you do not use a proxy server, do not specify any values for the `proxyserverusername`, `proxyserverpassword`, and `proxyserverdomain` parameters.

```
import java.text.SimpleDateFormat

String username = 'service_administrator'
String password = 'examplePWD'
String url = 'example_EPM_URL'
String updateversions = '01,04,07,10'
String podtype = 'test'
String proxyserverusername = ''
```

```
String proxyserverpassword = ''
String proxyserverdomain = ''

def currentdate = new Date()
def yf = new SimpleDateFormat("yy")
def mf = new SimpleDateFormat("MM")
String[] monthsarr = ["01", "02", "03", "04", "05", "06", "07", "08", "09",
"10", "11", "12"]
List<String> monthsarrfromcurrent = new ArrayList<>()
List<String> yearsarrfromcurrent = new ArrayList<>()
String currentyear = yf.format(currentdate)
String nextyear = (currentyear.toInteger() + 1).toString()
String currentmonth = mf.format(currentdate)

String[] updateVersionsStringArr = updateversions.split(',');
def updateversionsarr = new int[updateVersionsStringArr.length];
for(int i = 0; i < updateVersionsStringArr.length; i++)
{
    updateversionsarr[i] = Integer.parseInt(updateVersionsStringArr[i]);
}

def LogMessage(String message) {
    def date = new Date()
    def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss")
    println('[' + sdf.format(date) + '][GROOVY] ' + message);
}

def LogOperationStatus(EpmAutomateStatus opstatus) {
    def returncode = opstatus.getStatus()
    LogMessage(opstatus.getOutput())
    LogMessage('return code: ' + returncode)
}

int CompareUpdateMonths(int thismonth, int nextmonth) {
    int nextmonthorig = nextmonth

    if (nextmonth < thismonth) {
        nextmonth = nextmonth + 12
    }

    int monthdiff = nextmonth - thismonth

    if (monthdiff > 4) {
        LogMessage('There are more than 3 versions skipped from version ' +
thismonth + ' to version ' + nextmonthorig + '. Please correct updateversions
so that there are not more than three versions skipped between each update
version. Exiting.')
        return 1
    }

    return 0
}

int ValidateUpdateMonths(int[] updateversionsarr) {
    for(int i = 0; i < updateversionsarr.length; i++)
    {
```

```
        int nextint = i + 1
        String nextupdatemonth = ""
        int nextupdatemonthint = 0
        String thisupdatemonth = updateversionsarr[i]
        int thisupdatemonthint = thisupdatemonth.toInteger()

        if (nextint < updateversionsarr.length) {
            nextupdatemonth = updateversionsarr[nextint]
        } else {
            nextupdatemonth = updateversionsarr[0]
        }

        nextupdatemonthint = nextupdatemonth.toInteger()

        int returncode = CompareUpdateMonths(thisupdatemonthint,
nextupdatemonthint)
        if (returncode > 0) {
            return 1
        }
    }
    return 0
}

def SkipUpdateAdd(EpmAutomate automate, String yearnumber, String
monthnumber) {
    String yeardotmonth = yearnumber + '.' + monthnumber
    LogMessage('Running: epmautomate skipUpdate add version=' + yeardotmonth
+ ' comment=\"adding skipUpdate\"')
    EpmAutomateStatus status = automate.execute('skipupdate','add','version='
+ yeardotmonth,'comment=\"adding skipUpdate\"')
    LogOperationStatus(status)
}

LogMessage('Starting skip update processing')
EpmAutomate automate = getEpmAutomate()

// validate update months
int returncode = ValidateUpdateMonths(updateversionsarr)
if (returncode != 0) {
    return 1
}

// populate arrays
int startposition = 0
for(int i = 0; i < monthsarr.length; i++)
{
    if (currentmonth == monthsarr[i]) {
        if (podtype.equals("prod")) {
            if (updateVersionsStringArr.contains(currentmonth)) {
                startposition = (i-2)
            } else {
                startposition = (i-1)
            }
        } else {
            if (updateVersionsStringArr.contains(currentmonth)) {
                startposition = i
```

```
            } else {
                startposition = (i-1)
            }
        }

        break
    }
}

if (startposition < 0) {
    startposition = startposition + 12
}

for(int i = 0; i < monthsarr.length; i++)
{
    if (i >= startposition) {
        monthsarrfromcurrent.add(monthsarr[i])
        yearsarrfromcurrent.add(currentyear)
    }
}

for(int i = 0; i < monthsarr.length; i++)
{
    if (i <= startposition) {
        monthsarrfromcurrent.add(monthsarr[i])
        yearsarrfromcurrent.add(nextyear)
    }
}

// process skip updates
LogMessage("Operation: encrypt " + password + " oracleKey password.epw")
EpmAutomateStatus status =
automate.execute('encrypt',password,"oracleKey","password.epw")
LogOperationStatus(status)

if ((proxyserverusername != null && proxyserverusername != '') &&
(proxyserverpassword != null && proxyserverpassword != '') &&
(proxyserverdomain != null && proxyserverdomain != '')) {
    LogMessage("Operation: login " + username + " password.epw " + url + "
ProxyServerUserName=" + proxyserverusername + " ProxyServerPassword=" +
proxyserverpassword + " ProxyServerDomain=" + proxyserverdomain)
    status =
automate.execute('login',username,"password.epw",url,"ProxyServerUserName=" +
proxyserverusername,"ProxyServerPassword=" +
proxyserverpassword,"ProxyServerDomain=" + proxyserverdomain)
    LogOperationStatus(status)
} else {
    LogMessage("Operation: login " + username + " password.epw " + url)
    status = automate.execute('login',username,"password.epw",url)
    LogOperationStatus(status)
}
LogMessage('Running: epmautomate skipUpdate remove')
status = automate.execute('skipupdate','remove')
LogOperationStatus(status)

int addcount = 0
```

```
int countlimit = 0

if (podtype.equals("prod")) {
    countlimit = 3
} else {
    countlimit = 2
}

for (int i = 0; i < monthsarrfromcurrent.size(); i++) {
    int match = 1

    if (addcount == countlimit){
        LogMessage('Update add calls are completed. No more will be
attempted.')
        break
    }

    for(int j = 0; j < updateversionsarr.length; j++) {

        if ((Integer.parseInt(currentmonth) == updateversionsarr[j]) &&
(addcount > 0)) {
            match = 1
            break
        }

        if ((Integer.parseInt(monthsarrfromcurrent.get(i)) ==
updateversionsarr[j]) && (addcount == 0)) {
            match = 0
            break
        }
    }

    if (match == 1) {
        SkipUpdateAdd(automate, yearsarrfromcurrent.get(i),
monthsarrfromcurrent.get(i))
        addcount+=1
    }
}

LogMessage('Running: epmautomate skipUpdate list')
status = automate.execute('skipupdate','list')
LogOperationStatus(status)
println(status.getItemsList())

LogMessage('Running: epmautomate logout')
status = automate.execute('logout')
LogOperationStatus(status)

LogMessage('Skip update processing completed')
```

Creating the input.properties File to Run `skip_update` Windows and Linux/UNIX Scripts

To run `skip_update.ps1` or `skip_update.sh`, create the `input.properties` file and update it with information for your environment. Save the file in a local directory. Contents of this file differ depending on your operating system.

Windows

```
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updateversions=01,04,07,10
podtype=test
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updatemonths=02,05,08,11
```

Table 3-14 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`url`	URL of the environment on which you want to set the non-monthly update cadence.
`updateversions`	`updateversions` A comma separated list of EPM Cloud updates that should be applied to the environment identified by the `url` parameter. For example, `updateversions=02,05,08,11`. Versions must be specified as two digits; include a preceding zero for updates 01 through 09. The script attempts to run the skipUpdate command for the updates not included in the `updateversions` parameter value. For example, if you specify `updateversions=02,05,08,11`, the script tries to set skip update flags for the 01 (January), 03 (March), 04 (April), 06 (June), 07 (July), 09 (September), 10 (October), and 12 (December) updates. In this case, EPM Cloud updates 02 (February), 05 (May), 08 (August), and 11 (November) are applied to the environment.
`podtype`	EPM Cloud environment type. Valid values are `test` and `prod`.

Running the Script

1. **For Windows and Linux/UNIX: only**
   - Create `skip_update.ps1` or `skip_update.sh` by copying the script from a preceding section.
   - Create the `input.properties` file and save it in the directory where `skip_update` script is located. Contents of this file differs depending on your operating system. See

Creating the input.properties File to Run skip_update Windows and Linux/UNIX Scripts.
Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.

- Launch the script.
  - **Windows PowerShell:** run `skip_update.ps1`.
  - **Linux/UNIX:** run `./skip_update.sh`.

2. **Server-Side Groovy:**
   - Create the `skip_update.groovy` Groovy script and update it as required.
   - Use the Groovy screen in an EPM Cloud business process or automate the script execution using runBusinessRule. For information on running Groovy Script using EPM Automate, see Running Commands without Installing EPM Automate.

Sample Scenarios for Planning, Consolidation, Tax Reporting, and Enterprise Profitability and Cost Management

The scripts available in this section helps you automate tasks in Planning (including Planning Modules), Financial Consolidation and Close, Tax Reporting, and Enterprise Profitability and Cost Management environments.

Related Topics

- Automate the Export of a Large Number of Cells from an Aggregate Storage Cube
  Use the PowerShell or Bash script in this section to export a large number of cells from an Aggregate Storage (ASO) cube.
- Import Metadata into an Application
  Use these scripts to manually import application metadata from a file.
- Import Data, Run a Calculation Script, and Copy Data from a Block Storage Database to an Aggregate Storage Database
  Use these scripts to import data from a file, refresh the cube, run a business rule to calculate the cube, and then push data to an ASO cube.
- Export and Download Metadata and Data
  Use these scrips tp to export application metadata and data, and then to download the export files to a local directory.
- Export and Download Application Data
  Use these scripts to export application data and then to download it to a local directory.
- Automate the Archiving of Application Audit Records
  Use the Windows and Linux scripts in this section to automate the process of exporting and archiving application audit data to a local computer.
- Upload a Data File to an Environment and Run a Data Load Rule
  Use these scripts to upload a file to an environment and then run a data rule to import data from the file into an application.
- Automate Daily Data Integration
  This scenario explores the use of a sample script to automate data integration on a regular basis.

Automate the Export of a Large Number of Cells from an Aggregate Storage Cube

Use the PowerShell or Bash script in this section to export a large number of cells from an Aggregate Storage (ASO) cube.

Because the limits imposed by Oracle Essbase `QUERYRESULTLIMIT`, it is impossible to export a large quantity of data from the user interface. The PowerShell script available in this section splits the export operation into a specified number of jobs, run each job, downloads the exported data, and concatenates the export files into one export file ensuring that only one header is present.

> **Note:**
>
> These scripts execute an existing job of type export data. For detailed instructions on creating jobs, see "Managing Jobs" in *Administering Planning*.

PowerShell Script

```
$user = '<USERNAME>'
$pass = '<PASSWORD>'
$serverURL = '<URL>'
$applicationName = '<APPLICATIONNAME>'
$cubeName = '<CUBENAME>'
$splitDimension = '<DIMENSION_TO_SPLIT_THE_EXPORT>'
$topLevelMemberForExport = '<TOP_MEMBER_FOR_EXPORT>'
$exportJobName = '<EXPORT_JOB_NAME>'
$exportFilePrefix = '<PREFIX_FOR_EXPORT_FILE>'
$columnMembers = '<MEMBERS_ON_COLUMNS>'
$povMembers = '<POV_MEMBERS>'
$numberOfExportFiles = <NUMBER_OF_FILES_TO_SPLIT_THE_EXPORT>

$memberArray = @()
$exportFileArray = @()

function getLevel0 ($parent) {
    $parent.children.ForEach({
        if ( $_.children.count -eq 0 ) {
            $script:memberArray += $_.name
        }
        getLevel0($_)
    })
}

function findMember ($tree, $memberName) {
    $subtree = ""
    if ($tree.name -eq $memberName){
        return $tree
    } else {
        $tree.children.ForEach({
            #Write-Host $_.name
            if ($subtree -eq ""){ $subtree = findMember $_ $memberName}
```

```
        })
        return $subtree
    }
}

#putting together base64 encoded authentication header based un user and
password
$encodedCredentials =
[Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($($user) +
":" + $($pass)))
$headers = @{ Authorization = "Basic $encodedCredentials" }

#test login
$testRequest = $serverURL + '/HyperionPlanning/rest/v3/applications'

try {
    $response = Invoke-RestMethod -Uri $testRequest -Method Get -
Headers $headers -UseBasicParsing
}
catch {
    Write-Host $_
    return
}

#retrieve dimension hierarchy from application
Write-Host "Retrieving member list for split dimension " $splitDimension
$request = $serverURL + '/HyperionPlanning/rest/v3/internal/applications/'
+ $applicationName + '/plantypes/' + $cubeName + '/dimensions/'
+ $splitDimension
try {
    $response = Invoke-RestMethod -Uri $request -Method Get -Headers $headers
-UseBasicParsing
}
catch {
    Write-Host $_
    return
}
Write-Host $splitDimension " member list retrieved"

#search for the top of the export hierarchy
Write-Host "Searching for member " $topLevelMemberForExport " in hierarchy"
$member = findMember $response $topLevelMemberForExport
if ( $member.name -ne $topLevelMemberForExport ) {
    Write-Host $topLevelMemberForExport " not found in hierarchy, exiting ..."
    return 128
}
Write-Host "Found member " $topLevelMemberForExport " in hierarchy"

#retrieve level 0 memebers in export hierarchy
Write-Host "Retrieving Level 0 members for hierarchy"
getLevel0($member)
if ( $memberArray.Length -eq 0 ) {
    Write-Host "no level 0 members found in hierarchy, exiting ..."
    return 128
}
Write-Host $memberArray.Length " Level 0 members for export hierarchy
```

```
retrieved"


$request = $serverURL + '/HyperionPlanning/rest/v3/applications/'
+ $applicationName + '/jobs'

#splitting member list into the number of export files
$numberOfEntitiesPerFile =
[math]::truncate($memberArray.Length / $numberOfExportFiles)
for ($i = 1; $i -le $numberOfExportFiles; $i++) {
    $memberList = ""
    $firstMember = ($i - 1) * $numberOfEntitiesPerFile
    if ($i -lt $numberOfExportFiles) {
        $lastMember = $i * $numberOfEntitiesPerFile
    } else {
        $lastMember = $i * $numberOfEntitiesPerFile + $memberArray.Length
% $numberOfExportFiles
    }
    for ($j = $firstMember; $j -lt $lastMember; $j++) {
        $memberList += $memberArray[$j]
        if ($j -lt $lastMember - 1) {$memberList += ","} #avoid adding a
comma (,) after the last member of each set
    }

    $jobDetails='
    {
    "jobType":"EXPORT_DATA","jobName":"' + $exportJobName + '",
    "parameters":{
                  "exportFileName":"Export-' + $i + '.zip",
                  "rowMembers":"' +  $memberList + '",
                  "columnMembers":"' + $columnMembers + '",
                "povMembers":"' + $povMembers + '"
                 }
    }'

    #start export job
    try{
        $response = Invoke-RestMethod -Uri $request -Method Post -
Headers $headers -Body $jobDetails -ContentType "application/json"}
    catch {
        Write-Host $_
        return
    }

    Write-Host "Started export job " $i " out of " $numberOfExportFiles

    #checking job status, continue once jos is completed
    $statusRequest = $serverURL + '/HyperionPlanning/rest/v3/applications/'
+ $applicationName + '/jobs/' + $response.jobId
    $statusResponse = Invoke-RestMethod -Uri $statusRequest -Method Get -
Headers $headers -UseBasicParsing

    while ( $statusResponse.descriptiveStatus -eq "Processing" ) {
        Write-Host $statusResponse.descriptiveStatus
        Start-Sleep -s 10
        $statusResponse = Invoke-RestMethod -Uri $statusRequest -Method Get -
```

```
Headers $headers -UseBasicParsing
    }
    Write-Host $statusResponse.descriptiveStatus

    Write-Host "Downloading export file ..."
    $downloadRequest = $serverURL + '/interop/rest/11.1.2.3.600/
applicationsnapshots/Export-' + $i + '.zip/contents'
    $statusResponse = Invoke-RestMethod -Uri $downloadRequest -Method Get -
Headers $headers -OutFile "$exportFilePrefix-$i.zip"

    Write-Host "Expanding archive ..."
    Expand-Archive -Force -LiteralPath "$exportFilePrefix-$i.zip" -
DestinationPath "$exportFilePrefix-$i"
    Remove-Item "$exportFilePrefix-$i.zip"

    Get-ChildItem -Path "$exportFilePrefix-$i" -File -Name | ForEach-Object
{ $exportFileArray += "$exportFilePrefix-$i\" +  $_ }
}

Write-Host "creating outputfile ..."
#write header to outputfile
Get-Content $exportFileArray[0] | Select-Object -First 1 | Out-File
"$exportFilePrefix.csv"

#write content to outputfile skipping header
ForEach ($exportFile in $exportFileArray) {
    Get-Content $exportFile | Select-Object -Skip 1 | Out-File -Append
"$exportFilePrefix.csv"
}

Compress-Archive -LiteralPath "$exportFilePrefix.csv"  -DestinationPath
"$exportFilePrefix.zip"

Write-Host "cleaning up ..."
Remove-Item "$exportFilePrefix-*" -Recurse
Remove-Item "$exportFilePrefix.csv"
```

Bash Script

```
#!/bin/bash

user='<USERNAME>'
pass='<PASSWORD>'
serverURL='<URL>'
applicationName='<APPLICATIONNAME>'
cubeName='<CUBENAME>'
splitDimension='<DIMENSION_TO_SPLIT_THE_EXPORT>'
topLevelMemberForExport='<TOP_MEMBER_FOR_EXPORT>'
exportJobName='<EXPORT_JOB_NAME>'
exportFilePrefix='<PREFIX_FOR_EXPORT_FILE>'
columnMembers='<MEMBERS_ON_COLUMNS>'
povMembers='<POV_MEMBERS>'
numberOfExportFiles=<NUMBER_OF_FILES_TO_SPLIT_THE_EXPORT>

getRowMembers() {
```

```
    local memberList="$1"
    local firstMember=$2
    local lastMember=$3
    local nameCount=0
    local rowMember=""
    local rowMembers=""

    while IFS= read -r line
    do
        if [[ "${line}" == *"name"* ]]
        then
            if [[ ${nameCount} -ge ${firstMember} ]] && [[ ${nameCount} -lt $
{lastMember} ]]
            then
              rowMember=$(echo "${line}" | cut -d':' -f2- | sed s'/[",]//g')
              rowMembers="${rowMembers}${rowMember},"
            fi
            ((nameCount+=1))
        fi
    done <<< "${memberList}"
    rowMembers=$(echo "${rowMembers}" | rev | cut -d',' -f2- | rev)
    echo "${rowMembers}"
}

getLevel0()
{
    local memberList="$1"
    local names=$(echo "${memberList}" | jq 'recurse (try .children[])
| .name' | sed -e 's/"//g')
    local elements=""

    formerIFS=$IFS
    IFS=$'\n'
    namesarr=($names)
    IFS=$formerIFS

    for i in ${!namesarr[@]}
    do
        testelement=$(echo "${memberList}" | jq --arg currentName "$
{namesarr[i]}" 'recurse (try .children[]) | select(.name==$currentName)')
        if [[ "${testelement}" != *"children"* ]]
        then
            elements="${elements}${testelement}"
        fi
    done

    echo "${elements}"
}

#test login
header="Content-Type: application/x-www-form-urlencoded"
applicationsRequest="${serverURL}/HyperionPlanning/rest/v3/applications"
response=$(curl -X "GET" -s -w "%{http_code}" -u "${user}:${pass}" -H "$
{header}" "${applicationsRequest}")
http_response_code=$(echo "${response}" | rev | cut -d'}' -f1 | rev)
```

```
if [ ${http_response_code} -ne 200 ]
then
    echo "${response}"
    exit
fi

#retrieve dimension hierarchy from application
echo "Retrieving member list for split dimension ${splitDimension}"
splitDimensionRequest="${serverURL}/HyperionPlanning/rest/v3/internal/
applications/${applicationName}/plantypes/${cubeName}/dimensions/$
{splitDimension}"
response=$(curl -X GET -s -w "%{http_code}" -u "${user}:${pass}" -o "response-
memberlist.txt" -D "respHeader-memberlist.txt" -H "${header}" "$
{splitDimensionRequest}")
http_response_code=$(echo "${response}" | rev | cut -d'}' -f1 | rev)

if [ ${http_response_code} -ne 200 ]
then
    echo "${response}"
    exit
fi

echo "${splitDimension} member list retrieved"

#search for the top of the export hierarchy
echo "Searching for member ${topLevelMemberForExport} in hierarchy"
memberList=$(cat response-memberlist.txt | jq --arg topLevelMember "$
{topLevelMemberForExport}" 'recurse(try .children[]) | select (.name
== $topLevelMember)')
if [[ "${memberList}" == "" ]]
then
    echo "${topLevelMemberForExport} not found in hierarchy, exiting ..."
    exit 128
fi

echo "Found member ${topLevelMemberForExport} in hierarchy"

#retrieve level 0 members in export hierarchy
echo "Retrieving Level 0 members for hierarchy"
totalCount=$(echo "${memberList}" | grep "name" | wc -l)
grepChildrenCount=$(echo "${memberList}" | grep "children" | wc -l)
levelZeroCount=$((totalCount-grepChildrenCount))

if [[ "${levelZeroCount}" -eq 0 ]]
then
    echo "no level 0 members found in hierarchy, exiting ..."
    exit 128
fi

echo "${levelZeroCount} Level 0 members for export hierarchy retrieved"

#splitting member list into the number of export files
numberOfEntitiesPerFile=$((levelZeroCount/numberOfExportFiles))
jobsRequest="${serverURL}/HyperionPlanning/rest/v3/applications/$
{applicationName}/jobs"
header="Content-Type: application/json"
```

```
for ((i = 1 ; i <= ${numberOfExportFiles}; i++))
do
    firstMember=$((($i-1)*numberOfEntitiesPerFile))
    if [[ ${i} -lt ${numberOfExportFiles} ]]
    then
        lastMember=$((i*numberOfEntitiesPerFile))
    else
        lastMember=$
((i*numberOfEntitiesPerFile+levelZeroCount%numberOfExportFiles))
    fi

    elements=$(getLevel0 "${memberList}")
    rowMembers=$(getRowMembers "${elements}" ${firstMember} ${lastMember})

    response=$(curl -X POST -s -w "%{http_code}" -u "${user}:${pass}" -o
"response-job.txt" -D "respHeader-job.txt" -H "${header}" "${jobsRequest}" -d
'{"jobType":"EXPORT_DATA","jobName":"'"${exportJobName}"'","parameters":
{"exportFileName":"Export-'"${i}"'.zip","rowMembers":"'"$
{rowMembers}"'","columnMembers":"'"${columnMembers}"'","povMembers":"'"$
{povMembers}"'"}}')

    echo "Started export job " $i " out of " $numberOfExportFiles
    jobId=$(cat response-job.txt | grep -o '"jobId":[^, }]*' | cut -d':' -f2)
    descriptiveStatus=$(cat response-job.txt | grep -o '"descriptiveStatus":
[^, }]*' | cut -d':' -f2 | sed -e 's/"//g')
    jobIdRequest="${serverURL}/HyperionPlanning/rest/v3/applications/$
{applicationName}/jobs/${jobId}"
    response=$(curl -X GET -s -w "%{http_code}" -u "${user}:${pass}" -o
"response-jobstatus.txt" -D "respHeader-jobstatus.txt" -H "${header}" "$
{jobIdRequest}")

    jobId=$(cat response-jobstatus.txt | grep -o '"jobId":[^, }]*' | cut -
d':' -f2)
    descriptiveStatus=$(cat response-jobstatus.txt | grep -o
'"descriptiveStatus":[^, }]*' | cut -d':' -f2 | sed -e 's/"//g')

    while [[ "${descriptiveStatus}" == "Processing" ]]
    do
        echo "${descriptiveStatus}"
        sleep 10
        response=$(curl -X GET -s -w "%{http_code}" -u "${user}:${pass}" -o
"response-jobstatus.txt" -D "respHeader-jobstatus.txt" -H "${header}" "$
{jobIdRequest}")
        descriptiveStatus=$(cat response-jobstatus.txt | grep -o
'"descriptiveStatus":[^, }]*' | cut -d':' -f2 | sed -e 's/"//g')
    done

    echo "${descriptiveStatus}"

    echo "Downloading export file ..."
    contentsRequest="${serverURL}/interop/rest/11.1.2.3.600/
applicationsnapshots/Export-${i}.zip/contents"
    curl -X GET -s -w "%{http_code}" -u "${user}:${pass}" -D "respHeader-
download.txt" "${contentsRequest}" > "${exportFilePrefix}-${i}.zip"
```

```
    echo "Expanding archive ..."
    unzip "${exportFilePrefix}-${i}.zip" -d "${exportFilePrefix}-${i}"
    rm "${exportFilePrefix}-${i}.zip"

    echo "Writing to outputfile ..."
    if [[ -d "${exportFilePrefix}-${i}" ]]
    then
        find "${exportFilePrefix}-${i}" -name \*.csv | xargs cat | tail -n +2
>> "${exportFilePrefix}.csv"
    fi
done
zip "${exportFilePrefix}.zip" "${exportFilePrefix}.csv"

echo "cleaning up ..."
find . -name "${exportFilePrefix}-*" | xargs rm -r
rm "${exportFilePrefix}.csv"
```

To export a large number of cells from an Aggregate Storage (ASO) cube:

1. Copy the PowerShell or Bash script and save it to your file system, for example, as `ASOCellExport.ps1` or `ASOCellExport.sh`.
2. Modify the script file and set parameter values. See the following table for details.

Table 3-15 Variable Values to Include in the PowerShell and Bash Scripts

Variable	Description
`user`	Domain and user name of a Service Administrator in *`DOMAIN.USER`* format. **Examples:** **Windows:** `$user = 'exampleDomain.jDoe'` **Linux/UNIX:** `user = 'exampleDomain.jDoe'`
`pass`	Password of the Service Administrator or the location of the encrypted password file. See the encrypt command for information on creating an encrypted password file. **Examples:** **Windows:** `$pass = 'Example'` **Linux/UNIX:** `pass = 'Example'`
`serverURL`	The URL of the Oracle Enterprise Performance Management Cloud environment. **Examples:** **Windows:** `$serverURL = 'https://example .oraclecloud.com'` **Linux/UNIX:** `serverURL = 'https://example .oraclecloud.com'`
`applicationName`	Name of a Planning, Financial Consolidation and Close, Tax Reporting or Enterprise Profitability and Cost Management application. **Examples:** **Windows:** `$applicationName = 'Vision'` **Linux/UNIX:** `applicationName = 'Vision'`

Table 3-15 (Cont.) Variable Values to Include in the PowerShell and Bash Scripts

Variable	Description
cubeName	Name of a Cube in the application. **Examples:** **Windows:** `$cubeName = 'VisASO'` **Linux/UNIX:** `cubeName = 'VisASO'`
splitDimension	Name of a dimension the members of which are used to split the export into groups. **Examples:** **Windows:** `$splitDimension = 'Account'` **Linux/UNIX:** `splitDimension = 'Account'`
topLevelMemberForExport	Name of a member of the dimension sub-hierarchy under which a list of Level 0 members is created. **Examples:** **Windows:** `$topLevelMemberForExport = 'Total Cash Flow'` **Linux/UNIX:** `topLevelMemberForExport = 'Total Cash Flow'`
exportJobName	Name of an existing job of type Export Data. The settings specified in this job will be overwritten by the parameters that you set in the script. **Examples:** **Windows:** `$exportJobName = 'ASO Cell Export'` **Linux/UNIX:** `exportJobName = 'ASO Cell Export'`
exportFilePrefix	A file name prefix to uniquely identify the files generated by the export job. **Examples:** **Windows:** `$exportFilePrefix = 'cashflow'` **Linux/UNIX:** `exportFilePrefix = 'cashflow'`
columnMembers	The member columns to include in the export. **Examples:** **Windows:** `$columnMembers = 'Period'` **Linux/UNIX:** `columnMembers = 'Period'`

povMembers	Point of Views to include in the export. POV Members must include all other dimensions and can include functions as shown below:
	`ILvl0Descendants(YearTotal),`
	`ILvl0Descendants(Year),`
	`ILvl0Descendants(Scenario),`
	`ILvl0Descendants(Version),`
	`ILvl0Descendants(P_TP), ILvl0Descendants(AltYear)`
	Examples:
	Windows: `$povMembers = 'YTD'`
	Linux/UNIX: `povMembers = 'YTD'`

Table 3-15 (Cont.) Variable Values to Include in the PowerShell and Bash Scripts

Variable	Description
`numberOfExportFiles`	Number of jobs to execute for this export operation. If export still fails due to query limit limitations, increase this number. **Examples:** **Windows:** `$numberOfExportFiles = 3` **Linux/UNIX:** `numberOfExportFiles = 3`

3. Using Windows Scheduler or a cron job, schedule the script to execute at a convenient time. See Automating Script Execution for detailed steps.

Import Metadata into an Application

Use these scripts to manually import application metadata from a file.

These scripts perform the following activities:

- Signs in to an environment.
- Uploads a metadata file.
- Imports metadata from the uploaded file into the application using a job.
- Refreshes the cube.
- Signs out.

Windows Sample Script

Create `importMetadata.ps1` by copying the following Script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$file1="$($inputproperties.file1)"
$jobName="$($inputproperties.jobName)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${file1}
epmautomate importmetadata ${jobName} ${file1}
epmautomate refreshcube
epmautomate logout
```

Linux/UNIX Sample Script

Create `importMetadata.sh` by copying the following Script. Store it in a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${file1}"
${epmautomatescript} importmetadata "${jobName}" "${file1}"
```

```
${epmautomatescript} refreshcube
${epmautomatescript} logout
```

Creating the input.properties File

Create the `input.properties` file by copying one of the following and updating it with information for your environment. Save the file in the directory where `importMetadata.ps1` or `importMetadata.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
File1=FILE_NAME.zip
jobName=JOB_NAME
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
File1=FILE_NAME.zip
jobName=JOB_NAME
```

Table 3-16 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator, who also has the Identity Domain Administrator role.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment from which you want to generate the snapshot.
`File1`	Name of the ZIP file containing the metadata to import.
`JobName`	Job to use for importing the metadata.

Running the Scripts

1. Create `importMetadata.ps1` or `importMetadata.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `importMetadata` script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.

3. Launch the script.
   - **Windows PowerShell:** run `importMetadata.ps1`.
   - **Linux/UNIX:** run `./importMetadata.sh`.

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

Use these scripts to import data from a file, refresh the cube, run a business rule to calculate the cube, and then push data to an ASO cube.

These scripts perform the following actions:

- Signs in to an environment.
- Uploads a file `data.csv`.
- Imports data from `data.csv` into the application using job `loadingq1data`.
- Refreshes the cube.
- Runs business rules to transform data.
- Pushes data to an aggregate storage database using a job.
- Signs out.

Windows Sample Script

Create `importDataPlus.ps1` by copying this script. Save it to a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$importDataJobName="$($inputproperties.importDataJobName)"
$businessRuleName="$($inputproperties.businessRuleName)"
$planTypeMapName="$($inputproperties.planTypeMapName)"
$param1Key="$($inputproperties.param1Key)"
$param1Value="$($inputproperties.param1Value)"
$param2Key="$($inputproperties.param2Key)"
$param2Value="$($inputproperties.param2Value)"
$clearData="$($inputproperties.clearData)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${file1}
epmautomate importdata ${importDataJobName} ${file1}
epmautomate refreshcube
epmautomate runbusinessrule ${businessRuleName} ${param1Key}=${param1Value} $
{param2Key}=${param2Value}
epmautomate runplantypemap ${planTypeMapName} clearData=${clearData}
epmautomate logout
```

Linux/UNIX Sample Script

Create `importDataPlus.ps1` by copying this script. Save it to a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${file1}"
${epmautomatescript} importdata "${importDataJobName}" "${file1}"
${epmautomatescript} refreshcube
${epmautomatescript} runbusinessrule "${businessRuleName}" "${param1Key}=$
{param1Value}" "${param2Key}=${param2Value}"
${epmautomatescript} runplantypemap "${planTypeMapName}" clearData=$
{clearData}
${epmautomatescript} logout
```

Creating the input.properties File

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
File1=FILE_NAME.csv
importDataJobName=FILE_NAME
businessRuleName=RULE_NAME
planTypeMapName=PLAN_TYPE_MAP_NAME
param1Key=RUN-TIME PARAMETER_1
param1Value=RUN-TIME PARAMETER_1_VALUE
param2Key=RUN-TIME PARAMETER_2
param2Value=RUN-TIME PARAMETER_2_VALUE
clearData=true
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
File1=FILE_NAME.csv
importDataJobName=FILE_NAME
businessRuleName=RULE_NAME
planTypeMapName=PLAN_TYPE_MAP_NAME
param1Key=RUN-TIME PARAMETER_1
param1Value=RUN-TIME PARAMETER_1_VALUE
param2Key=RUN-TIME PARAMETER_2
param2Value=RUN-TIME PARAMETER_2_VALUE
clearData=true
```

Table 3-17 input.properties Parameters

Parameter	Description
javahome	JAVA_HOME location. For Linux/UNIX only.
epmautomatescript	Absolute path of EPM Automate executable (epmautomate.sh). For Linux/UNIX only.
username	User name of a Service Administrator, who also has the Identity Domain Administrator role.
password	Password of the Service Administrator or the name and location of the encrypted password file.
serviceURL	URL of the environment from which you want to generate the snapshot.
File1	Import file from which the data is to be loaded into the application.
iimportDataJobName	Name of the job to use for importing data.
businessRuleName	The business rule to run on the imported data
planTypeMapName	The plan type map to use for copying data from a BSO database to an ASO database or from a BSO database to another BSO database.
param1Key	Run-time prompt 1 to run the business rule.
param1Value	Value of run-time prompt 1.
param2Key	Run-time prompt 2 to run the business rule.
param2Value	Value of run-time prompt 2.
clearData	Indicates whether the data in the receiving database is to be deleted. Specify false to retain the data.

Running the Scripts

1. Create importDataPlus.ps1 or importDataPlus.sh by copying the script from a preceding section.
2. Create the input.properties file and save it in the directory where the importDataPlus script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run importDataPlus.ps1.
   - **Linux/UNIX:** run ./importDataPlus.sh.

Export and Download Metadata and Data

Use these scrips tp to export application metadata and data, and then to download the export files to a local directory.

These scripts complete the following activities:

- Signs in to an environment.
- Exports the metadata into a zip file using a specified job.
- Exports the application data into a zip file using a specified job.

- Lists the contents of the Inbox/Outbox.
- Downloads the exported data files to the local computer.
- Sign out.

Windows Sample Script

Create `exportDownloadMetadataAndData.ps1` by copying the following Script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$exportFile1="$($inputproperties.exportFile1)"
$exportFile2="$($inputproperties.exportFile2)"
$exportMetaDataJobName="$($inputproperties.exportMetaDataJobName)"
$exportDataJobName="$($inputproperties.exportDataJobName)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate exportmetadata ${exportMetaDataJobName} ${exportFile1}
epmautomate exportdata ${exportDataJobName} ${exportFile2}
epmautomate listfiles
epmautomate downloadfile ${exportFile1}
epmautomate downloadfile f${exportFile2}
epmautomate logout
```

Linux/UNIX Sample Script

Create `exportDownloadMetadataAndData.sh` by copying the following Script. Store it in a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} exportmetadata "${exportMetaDataJobName}" "$
{exportFile1}"
${epmautomatescript} exportdata "${exportDataJobName}" "${exportFile2}"
${epmautomatescript} listfiles
${epmautomatescript} downloadfile "${exportFile1}"
${epmautomatescript} downloadfile "${exportFile2}"
${epmautomatescript} logout
```

Creating the Properties File

Create the `input.properties` file by copying one of the following and updating it with information for your environment. Save the file in the directory where `exportDownloadMetadataAndData.ps1` or `exportDownloadMetadataAndData.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
exportFile1=FILE_NAME1.zip
```

```
exportFile2=FILE_NAME2.zip
exportMetaDataJobName=METADATA_EXPORT_JOB_NAME
exportDataJobName=DATA_EXPORT_JOB_NAME
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
exportFile1=FILE_NAME1.zip
exportFile2=FILE_NAME2.zip
exportMetaDataJobName=METADATA_EXPORT_JOB_NAME
exportDataJobName=DATA_EXPORT_JOB_NAME
```

Table 3-18 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator, who also has the Identity Domain Administrator role.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment from which you want to generate the snapshot.
`exportFile1`	Name of the file to which metadata is to be exported.
`exportFile2`	Name of the file to which adata is to be exported.
`exportDataJobName1`	Job to use for exporting the metadata.
`exportDataJobName2`	Job to use for exporting the data.

Running the Scripts

1. Create `exportDownloadMetadataAndData.ps1` or `exportDownloadMetadataAndData.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `exportDownloadMetadataAndData` script is located. Contents of this file differs depending on your operating system. See Creating the Properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run `exportDownloadMetadataAndData.ps1`.
   - **Linux/UNIX:** run `./exportDownloadMetadataAndData.sh`.

Export and Download Application Data

Use these scripts to export application data and then to download it to a local directory.

These scripts perform the following operations:

- Signs in to an environment.
- Backs up two sets of data using the jobs that you specify.
- Downloads the exported data files.
- Signs out.

Windows Sample Script

Create `exportDownloadData.ps1` by copying this script. Save it to a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$exportFile1="$($inputproperties.exportFile1)"
$exportFile2="$($inputproperties.exportFile2)"
$exportDataJobName1="$($inputproperties.exportDataJobName1)"
$exportDataJobName2="$($inputproperties.exportDataJobName2)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate exportdata ${exportDataJobName1} ${exportFile1}
epmautomate exportdata ${exportDataJobName2} ${exportFile2}
epmautomate listfiles
epmautomate downloadfile ${exportFile1}
epmautomate downloadfile ${exportFile2}
epmautomate logout
```

Linux/UNIX Sample Script

Create `exportDownloadData.sh` by copying this script. Save it to a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} exportdata "${exportDataJobName1}" "${exportFile1}"
${epmautomatescript} exportdata "${exportDataJobName2}" "${exportFile2}"
${epmautomatescript} listfiles
${epmautomatescript} downloadfile "${exportFile1}"
${epmautomatescript} downloadfile "${exportFile2}"
${epmautomatescript} logout
```

Creating the input.properties File

Create the `input.properties` file by copying one of the following and updating it with information for your environment. Save the file in the directory where `exportDownloadData.ps1` or `exportDownloadData.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
exportFile1=FILE_NAME.zip
exportFile2=FILE_NAME.zip
exportDataJobName1=JOB_NAME
exportDataJobName2=FILE_NAME
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
exportFile1=FILE_NAME.zip
exportFile2=FILE_NAME.zip
exportDataJobName1=FILE_NAME
exportDataJobName2=FILE_NAME
```

Table 3-19 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator, who also has the Identity Domain Administrator role.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment from which you want to generate the snapshot.
`exportFile1` and `exportFile2`	Name of the file to which data is to be exported.
`exportDataJobName1` and `exportDataJobName2`	Job to use for exporting the data.

Running the Scripts

1. Create `exportDownloadData.ps1` or `exportDownloadData.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `exportDownloadData` script is located. Contents of this file differs depending on your operating system. See Table 1.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run `exportDownloadData.ps1`.
   - **Linux/UNIX:** run `./exportDownloadData.sh`.

Automate the Archiving of Application Audit Records

Use the Windows and Linux scripts in this section to automate the process of exporting and archiving application audit data to a local computer.

Application audit data is retained for 365 days only. Customize these scripts and execute them once every 180 days, or as required by your data retention policies, to prevent the loss of historical audit data older than 365 days.

> **Note:**
>
> These scripts are tailored to archive data in local storage. You can modify them to archive the exported audit data files on network storage or in storage cloud (for example, Oracle Object Storage).

Table 3-20 Parameters and Their Values

Parameter	Value
`url`	The URL of the environment. **Examples:** • **Windows:** `set url=https://example-epmidm.epm.usphoenix-1.ocs.oraclecloud.com/epmcloud` • **Linux:** `url=https://example-epmidm.epm.usphoenix-1.ocs.oraclecloud.com/epmcloud`
`user`	The user name of a Service Administrator to sign into the environment to download audit data. **Examples:** • **Windows:** `set user=ExampleAdmin` • **Linux:** `user=ExampleAdmin`
`password`	Password of the Service Administrator (not recommended) or the name and location of the encrypted password file. See the encrypt command for information on creating an encrypted password file. **Examples:** • **Windows:** `set password="C:\mySecuredir\password.epw"` • **Linux:** `password="/home/user1/mySecuredir/password.epw"`
`AuditFileName`	Name of the audit data file. To make this file unique, the script appends the audit data export timestamp to this file name. **Example:** • **Windows:** `set AuditFileName=AuditData` • **Linux:** `AuditFileName=AuditData`
`NumberOfBackups`	Number of backup files to be retained in the storage. Default is 10, after which the oldest backup is replaced as needed. **Example:** • **Windows:** `set NumberOfBackups=20` • **Linux:** `NumberOfBackups=20`
For Linux script only	
`epmautomatescript`	The location where EPM Automate is installed. **Example:** `/home/user1/epmautomate/bin/epmautomate.sh`

Table 3-20 (Cont.) Parameters and Their Values

Parameter	Value
javahome	The *JAVA_HOME* location. **Example:** /home/user1/jdk1.8.0_191/

Windows Script

Create a batch file; for example, AuditExport.bat, containing script similar to the following to automate the exporting and downloading of audit data to a local computer.

```
@echo off
rem Sample script to download and maintain 10 audit data backups
rem Update the following  parameters

SET url=https://example.oraclecloud.com
SET user=ServiceAdmin
SET password=Example.epw
SET AuditFileName="AuditBackup"
SET NumberOfBackups=10

rem EPM Automate commands
call epmautomate login %user% %password% %url%
     IF %ERRORLEVEL% NEQ 0 goto :ERROR
       call epmautomate exportAppAudit %AuditFileName% nDays=180
     IF %ERRORLEVEL% NEQ 0 goto :ERROR
       call epmautomate downloadfile %AuditFileName%.zip
     IF %ERRORLEVEL% NEQ 0 goto :ERROR
       call epmautomate logout
     IF %ERRORLEVEL% NEQ 0 goto :ERROR

rem Rename downloaded audit data backup, keep the last 10 backups
Set Timestamp=%date:~4,2%_%date:~7,2%_%date:~10,2%%
Set Second=%time:~0,2%%time:~3,2%
ren %AuditFileName%.zip %AuditFileName%_%Timestamp%_%Second%.zip
SET Count=0
FOR %%A IN (%AuditFileName%*.*) DO SET /A Count += 1
IF %Count% gtr %NumberOfBackups% FOR %%A IN (%AuditFileName%*.*) DO del "%%A"
&& GOTO EOF
:EOF

echo Scheduled Task Completed successfully
exit /b %errorlevel%
:ERROR
echo Failed with error #%errorlevel%.
exit /b %errorlevel%
```

Linux Script

Create a shell script; for example, `AuditExport.sh`, containing script similar to the following to automate the exporting and downloading of audit data to a local computer.

```
#!/bin/sh
# Sample script to export, download and maintain 10 audit data backups
# Update the following seven parameters
url=https://example.oraclecloud.com
user=serviceAdmin
password=/home/user1/epmautomate/bin/example.epw
auditfilename="AuditBackup"
numberofbackups=10
epmautomatescript=/home/user1/epmautomate/bin/epmautomate.sh
javahome=/home/user1/jdk1.8.0_191/

export JAVA_HOME=${javahome}

printResult()
    {
    op="$1"
    opoutput="$2"
    returncode="$3"

    if [ "${returncode}" -ne 0 ]
    then
        echo "Command failed. Error code: ${returncode}. ${opoutput}"
    else
        echo "${opoutput}"
    fi
}

processCommand()
{
    op="$1"
    date=`date`

    echo "Running ${epmautomatescript} ${op}"
    operationoutput=`eval "$epmautomatescript $op"`
    printResult "$op" "$operationoutput" "$?"
}
op="login ${user} ${password} ${url}"
processCommand "${op}"
op="exportAppAudit \"${auditfilename}\" -nDays=180"
processCommand "${op}"
op="downloadfile \"${auditfilename}.zip\""
processCommand "${op}"
op="logout"
processCommand "${op}"
# Rename the downloaded audit data backup, keep the last 10 backups
timestamp=`date +%m_%d_%Y_%I%M`
mv "${auditfilename}.zip" "${auditfilename}_${timestamp}.zip"

((numberofbackups+=1))
```

```
ls -tp ${auditfilename}*.zip | grep -v '/$' | tail -n +${numberofbackups} |
xargs -d '\n' -r rm --
```

Upload a Data File to an Environment and Run a Data Load Rule

Use these scripts to upload a file to an environment and then run a data rule to import data from the file into an application.

Prerequisites

- The following definitions in Data Management:
  - A data load rule definition named `VisionActual`. It is assumed that the data rule does not specify a file path for the input file.
  - Period definitions `Mar-15` through `Jun-15`
- A properly formatted data file (`GLActual.dat`) that contains data.

To import data and run data load rule, you run commands that complete these steps:

- Sign in to the environment.
- Upload a file `GLActual.dat` that contains data for periods `Mar-15` through `Jun-15` into Data Management folder `inbox/Vision`.
- Import data from `GLActual.dat` into Data Management using data load rule `VisionActual`, start period `Mar-15`, end period `Jun-15`, and import mode `REPLACE`.
- Export data with the `STORE_DATA` option to merge the data in the Data Management staging table with existing application data.
- Sign out.

Windows Sample Script

Create `runDataLoadRule.ps1` by copying the following Script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$dataFile="$($inputproperties.dataFile)"
$dataRuleName="$($inputproperties.dataRuleName)"
$startPeriod="$($inputproperties.startPeriod)"
$endPeriod="$($inputproperties.endPeriod)"
$importMode="$($inputproperties.importMode)"
$exportMode="$($inputproperties.exportMode)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${datafile} ${dataFileUploadLocation}
epmautomate rundatarule ${dataRuleName} ${startPeriod} ${endPeriod} $
{importMode} ${exportMode} ${dataFileUploadLocation}/${dataFile}
epmautomate logout
```

Linux/UNIX Sample Script

Create `runDataLoadRule.sh` by copying the following Script. Store it in a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${datafile}" "${dataFileUploadLocation}"
${epmautomatescript} rundatarule "${dataRuleName}" "${startPeriod}" "$
{endPeriod}" "${importMode}" "${exportMode}" "${dataFileUploadLocation}/$
{dataFile}"
${epmautomatescript} logout
```

Creating the input.properties File

Create the `input.properties` file by copying one of the following and updating it with information for your environment. Save the file in the directory where `runDataLoadRule.ps1` or `runDataLoadRule.sh` is stored.

Windows

```
username=serviceAdmin
passwordfile=./password.epw
serviceURL=https://example.oraclecloud.com
dataFile=GLActual.dat
dataFileUploadLocation=UPLOAD_LOCATION
dataRuleName=RULE_NAME
startPeriod=START_PERIOD
endPeriod=END_PERIOD
importMode=IMPORT_MODE
exportMode=EXPORT_MODE
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURLdataFile=GLActual.dat
dataFileUploadLocation=UPLOAD_LOCATION
dataRuleName=RULE_NAME
startPeriod=START_PERIOD
endPeriod=END_PERIOD
importMode=IMPORT_MODE
exportMode=EXPORT_MODE
```

Table 3-21 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.

Table 3-21 (Cont.) input.properties Parameters

Parameter	Description
epmautomatescript	Absolute path of EPM Automate executable (epmautomate.sh). For Linux/UNIX only.
username	User name of a Service Administrator, who also has the Identity Domain Administrator role.
password	Password of the Service Administrator or the name and location of the encrypted password file.
serviceURL	URL of the environment from which you want to generate the snapshot.
dataFile	The file that contains the data to be imported using the data rule.
dataFileUploadLocation	Location to which the data file is to be uploaded.
dataRuleName	Name of a data load rule defined in Data Integration.
startPeriod	The first period for which data is to be loaded. This period name must be defined inData Integration period mapping.
endPeriod	For multi-period data load, the last period for which data is to be loaded. For single period load, use the same period as start period. This period name must be defined in Data Integration period mapping.
importMode	Mode for importing data into Data Integration. Use APPEND, REPLACE or RECALCULATE. Use NONE to skip data import into staging tables.
exportMode	Mode for exporting data to the application. Use Data Integration. Use STORE_DATA, ADD_DATA, SUBTRACT_DATA or REPLACE_DATA. Use NONE to skip data export from Data Integration to the application. **Note:** Financial Consolidation and Close supports only MERGE and NONE modes.

Running the Script

1. Create runDataLoadRule.ps1 or runDataLoadRule.sh by copying the script from a preceding section.
2. Create the input.properties file and save it in the directory where the runDataLoadRule script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run runDataLoadRule.ps1.
   - **Linux/UNIX:** run ./runDataLoadRule.sh.

Automate Daily Data Integration

This scenario explores the use of a sample script to automate data integration on a regular basis.

Create a batch (`.bat`) or shell (`.sh`) file that contains script similar to the following to automate data integration-related activities. The following sample script for Windows automates daily application data integration by completing these activities:

- Sign into an environment.
- Delete `DailyPlanData` if it is present.
- Upload `DailyPlanData` into the service.
- Run business rule `Clear Plan Targets` on plan type `Plan1`.
- Import data using job name `LoadDailyPlan`.
- Run business rule `Balance Sheet - Plan`.
- Run business rule `Allocate Plan Targets`.
- Delete `DailyTarget.zip` if it is present.
- Export data into `DailyTarget.zip` using job name `ExportDailyTarget`.
- Download `DailyTarget.zip` to your server and appends the timestamp.
- Sign out of the environment.

> **Note:**
>
> If you repurpose this script for your use, ensure that you modify the values of `SET url` and `SET user` parameters. Additionally, you may modify the values of `dataimportfilename`, `dataexportfilename`, `importdatajobname`, `exportdatajobname`, `br_clear`, `br_calculatebalancesheet`, and `br_allocatetarget` parameters to suit your requirements

See Automating Script Execution for information on scheduling the script using Windows Task Scheduler.

```
@echo off

rem Sample Script to demonstrate daily data integration with
rem EPM Cloud application.
rem This script uploads Plan data, clears target numbers,
rem runs a business rule to calculate balance sheet data, and
rem recalculates target numbers on the Vision demo application

rem Please update these parameters
SET url=https://example.oraclecloud.com
SET user=serviceAdmin
SET dataimportfilename=DailyPlanData.csv
SET dataexportfilename=DailyTarget
SET importdatajobname=LoadDailyPlan
SET exportdatajobname=ExportDailyTarget
SET br_clear=Clear Plan Targets
SET br_calculatebalancesheet=Balance Sheet - Plan
SET br_allocatetarget=Allocate Plan Targets

SET password=%1
```

```
rem Executing EPM Automate commands

CD /D %~dp0
call epmautomate login %user% %password% %url%
IF %ERRORLEVEL% NEQ 0 goto :ERROR

for /f %%i in ('call epmautomate listfiles') do if %%i==%dataimportfilename%
(call epmautomate deletefile %%i)
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate uploadfile %dataimportfilename%
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate runbusinessrule "%br_clear%"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate importdata "%importdatajobname%"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate runbusinessrule "%br_calculatebalancesheet%"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate runbusinessrule "%br_allocatetarget%"
"TargetVersion=Baseline"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

for /f %%i in ('call epmautomate listfiles') do if %
%i=="%dataexportfilename%.zip" (call epmautomate deletefile %%i)
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate exportdata %exportdatajobname% "%dataexportfilename%.zip"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

call epmautomate downloadfile "%dataexportfilename%.zip"
IF %ERRORLEVEL% NEQ 0 goto :ERROR

rem Section to rename the file

Set Timestamp=%date:~4,2%_%date:~7,2%_%date:~10,4%_%time:~1,1%%time:~3,2%%
ren "%dataexportfilename%.zip" "%dataexportfilename%_%Timestamp%.zip"

call epmautomate logout
IF %ERRORLEVEL% NEQ 0 goto :ERROR

:EOF
echo Scheduled Task Completed successfully
exit /b %errorlevel%

:ERROR
echo Failed with error #%errorlevel%.
exit /b %errorlevel%
```

Sample Scenarios for Account Reconciliation

Related Topics

- Load Preformatted Balances into a Period
  Use these scripts to import mapped data from an uploaded file into an Account Reconciliation environment.
- Upload and Import a Backup Snapshot
  Use these scripts to upload and import a backup snapshot into an Account Reconciliation environment.
- Archive Old Matched Transactions and Purge Archived Transactions
  Use the scripts in this section to archive matched transactions, including support and adjustment details, that are equal to or older than a specified age and then purge the archived transactions from Account Reconciliation. The archived matched transactions are recorded in a ZIP file.

Load Preformatted Balances into a Period

Use these scripts to import mapped data from an uploaded file into an Account Reconciliation environment.

Windows Sample Script

Create a file named `runPreformattedBalances.ps1` by copying the following script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$dataFile="$($inputproperties.dataFile)"
$period="$($inputproperties.period)"
$balanceType="$($inputproperties.balanceType)"
$currencyBucket="$($inputproperties.currencyBucket)"

$elements=$dataFile.split('/')
$dataFileName=$elements[-1]

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${dataFile}
epmautomate importpremappedbalances ${period} ${dataFileName} ${balanceType} $
{currencyBucket}
epmautomate deletefile ${dataFileName}
epmautomate logout
```

Linux/UNIX Sample Script

Create a file named `runPreformattedBalances.sh` by copying the following script. Store it in a local directory.

```
#!/bin/bash

. ./input.properties
```

```
export JAVA_HOME=${javahome}

dataFileName=$(echo "${dataFile}" | rev | cut -d'/' -f1 | rev)

${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${dataFile}"
${epmautomatescript} importpremappedbalances "${period}" "${dataFileName}" "$
{balanceType}" "${currencyBucket}"
${epmautomatescript} deletefile "${dataFileName}"
${epmautomatescript} logout
```

Sample input.properties File

To run the `runPreformattedBalances` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `runPreformattedBalances.sh` or `runPreformattedBalances.ps1` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
dataFile=DATA_FILE_NAME.csv
period=PERIOD_NAME
balanceType=BALANCE_TYPE
currencyBucket=CURRENCY_BUCKET
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
dataFile=DATA_FILE_NAME.csv
period=PERIOD_NAME
balanceType=BALANCE_TYPE
currencyBucket=CURRENCY_BUCKET
```

Table 3-22 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment that hosts the application into which you want to load preformatted balances.

Table 3-22 (Cont.) input.properties Parameters

Parameter	Description
`dataFile`	A CSV file that contains the preformatted balances (typically created from a General Ledger) that you want to load into the application. This file must already have been uploaded to the environment using the uploadFile command.
`period`	The reconciliation period to which the preformatted balances are to be uploaded.
`balanceType`	The type of preformatted balances contained in the `dataFile`.
`currencyBucket`	The currency bucket for the preformatted balances.

Running the Script

1. Create `runPreformattedBalances.ps1` or `runPreformattedBalances.sh` by copying the script from a preceding section.
2. **For Windows and Linux/UNIX: only**
   - Create the `input.properties` file and save it in the directory where the `runPreformattedBalances` script is located. Contents of this file differs depending on your operating system. See Table 1.
     Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
   - Launch the script.
     - **Windows PowerShell:** run `runPreformattedBalances.ps1`.
     - **Linux/UNIX:** run `./runPreformattedBalances.sh`.

Upload and Import a Backup Snapshot

Use these scripts to upload and import a backup snapshot into an Account Reconciliation environment.

Windows Sample Script

Create a file named `importBackupSnapshot.ps1` by copying the following script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$snapshotName="$($inputproperties.snapshotName)"
$userPassword="$($inputproperties.userPassword)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile ${snapshotName}.zip
epmautomate importsnapshot ${snapshotName} "userPassword=${userPassword}"
epmautomate deletefile ${snapshotName}.zip
epmautomate logout
```

Linux/UNIX Sample Script

Create a file named `importBackupSnapshot.sh` by copying the following script. Store it in a local directory

```
#!/bin/bash

. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${snapshotName}.zip"
${epmautomatescript} importsnapshot "${snapshotName}" "userPassword=$
{userPassword}"
${epmautomatescript} deletefile "${snapshotName}.zip"
${epmautomatescript} logout
```

Sample input.properties File

To run the `importBackupSnapshot` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `importBackupSnapshot.sh` or `importBackupSnapshot.ps1` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
snapshotName=SNAPSHOT_NAME
userPassword=IDM_NEW_USER_PWD
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
snapshotName=SNAPSHOT_NAME
userPassword=IDM_NEW_USER_PWD
```

Table 3-23 input.properties Parameters

Parameter	Description
`javahome`	`JAVA_HOME` location. For Linux/UNIX only.
`epmautomatescript`	Absolute path of EPM Automate executable (`epmautomate.sh`). For Linux/UNIX only.
`username`	User name of a Service Administrator.
`password`	Password of the Service Administrator or the name and location of the encrypted password file.
`serviceURL`	URL of the environment where you want to import the snapshot.
`snapshotName`	The name of the snapshot from which artifacts and data are to be imported. This snapshot must already have been uploaded to the environment using the uploadFile command.

Table 3-23 (Cont.) input.properties Parameters

Parameter	Description
`userPassword`	The default password that must be assigned to all new users created in the identity domain as a result of this snapshot import.

Running the Script

1. Create `importBackupSnapshot.ps1` or `importBackupSnapshot.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `runPreformattedBalances` script is located. Contents of this file differs depending on your operating system. See Sample input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run `importBackupSnapshot.ps1`.
   - **Linux/UNIX:** run `./importBackupSnapshot.sh`.

Archive Old Matched Transactions and Purge Archived Transactions

Use the scripts in this section to archive matched transactions, including support and adjustment details, that are equal to or older than a specified age and then purge the archived transactions from Account Reconciliation. The archived matched transactions are recorded in a ZIP file.

How the Script Works

1. Using the information in the `input.properties` file, logs into the environment
2. Runs the following `archiveTmTransactions` command to create an archive. Resulting ZIp file and log file use the defaulting names `Archive_Transactions_INTERCO_`*`JOB_ID`*`.zip` and `Archive_Transactions_INTERCO_`*`JOB_ID`*`.log`
   ```
   epmautomate archiveTmTransactions INTERCO 365 filterOperator=contains
   filterValue=14001
   ```
   You can change the command parameters by modifying the `input.properties` file.
3. Downloads the log file and the .ZIP file containing archived transactions to the local computer. The script displays an error message if no matching transactions are found.
4. Copies the .ZIP file containing archived transactions to Oracle Object Store.
5. Runs the `purgeArchivedTmTransactions` command (with the job ID of the archiveTmTransactions job) to delete the archived matched transactions from the application.

Running the Script

1. Create the `input.properties` file and update it with information for your environment. Save the file in a local, directory. This directory is referred to as `parentsnapshotdirectory` in this discussion. Contents of this file differs depending on your operating system.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run scripts.

2. Create `transaction_match.ps1` (Windows PowerShell) or `transaction_match.sh` (Linux/UNIX) script and save it in the `parentsnapshotdirectory` where `input.properties` is located.
3. Launch the script.
   - Linux/UNIX: run `./transaction_match.sh`.
   - Windows PowerShell: run `transaction_match.ps1`.

Creating the input.properties Script

Create `input.properties` by copying and updating the following script.

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
epmusername=exampleAdmin1
epmpassword=examplePassword1.epw
epmurl=exampleURL1
objectstorageusername=exampleAdmin2
objectstoragepassword=examplePassword2
objectstorageurl=exampleURL2
matchtype=INTERCO
age=365
filteroperator=contains
filtervalues=FilterValue=14001
proxyserverusername=myProxyserver
proxyserverpassword=myProxyserver_pwd
proxyserverdomain=myProxyDomain
```

> **Note:**
>
> **Windows only:** Remove these properties from the `input.properties` file:
>
> - `javahome=JAVA_HOME`
> - `epmautomatescript=EPM_AUTOMATE_LOCATION`
>
> If `authentication at proxy server` is not enabled for your Windows network environment, remove these properties from the `input.properties` file.
>
> - `proxyserverusername`
> - `proxyserverpassword`
> - `proxyserverdomain`

Table 3-24 input.properties Parameters

Parameter	Description
`javahome`	The directory where the JDK used by EPM Automate is installed. Delete this entry from the Windows version of `input.properties`. **Example:** `javahome=./home/JDK/bin`

Table 3-24 (Cont.) input.properties Parameters

Parameter	Description
epmautomatescript	The directory where EPM Automate is installed. Delete this entry from the Windows version of input.properties. **Example:**epmautomatescript=./home/utils/EPMAutomate/bin
epmusername	User name of a Service Administrator or a Power User, User, or Viewer who is authorized to archive matched transactions. **Example:** epmusername=ServiceAdmin
epmuserpassword	The encrypted password file for the user identified as epmusername. **Example:** epmpassword=myPwd.epw
epmurl	The URL of the environment wherein matched transactions are to be archived. **Example:** epmurl=https://test-cloudpln.pbcs.us1.oraclecloud.com
objectstorageusername	The ID of a user who has the required access rights to write to Oracle Object Storage Cloud. For users created in a federated identity provider, specify the fully-qualified name of the user (for example, exampleIdP/jdoe or exampleIdP/john.doe@example.com, where exampleIdP is the name of the federated identity provider). For other users, specify the User ID. **Example:** epmusername=myIdP/jdoe
objectstoragepassword	The Swift password or auth token associated with the user identified in objectstorageusername. This password is not the same as the password that the user uses to sign into the Object Storage Console. Auth token is an Oracle-generated token that you use to authenticate with third-party APIs, for example to authenticate with a Swift client. For instructions to create this token, see To create an auth token in *Oracle Cloud Infrastructure Documentation*. **Example:** objectstoragepassword=jDoe_PWD
objectstorageurl	The URL of the Oracle Object Storage Cloud bucket with an optional object name appended. **Example:** objectstorageurl=https://swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/MT_Archives/2023_archives
matchtype	The identifier (TextID) of the match type from which matched transactions should be archived. **Example:** matchtype=cashrecon
age	The number of days since the transaction was matched. Matched transaction older than or equal to this value will be archived. **Example:** age=180
filteroperator	The filter conditions to identify the accounts containing matched transactions for archival. Must be one of these: equals, not_equals, starts_with, ends_with, contains or not_contains. This value is combined with the filterValue to identify the accounts from which matched transactions are to be archived. **Example:** filteroperator=not_equals

Table 3-24 (Cont.) input.properties Parameters

Parameter	Description
filtervalues	One or more filter value to identify the transactions to archive. If equals or not_equals is specified as the filterOperator, you can use a space-separated list to specify multiple values. If multiple values are specified, transactions from accounts matching any filter operator and filter value combination are selected for archival. **Example:** filterValue=101-120 filterValue=140-202
proxyserverusername	The user name to authenticate a secure session with the proxy server that controls access to the internet. **Example:** proxyserverusername=myProxyserver
proxyserverpassword	The password to authenticate the user with the proxy server. **Example:** proxyserverpassword=myProxyserver_pwd
proxyserverdomain	The name of the domain defined for the proxy server. **Example:** proxyserverdomain=myProxyDomain

Windows Script

Create transaction_match.ps1 by copying the following script. Store it in the folder where input.properties is stored.

```
# Archive and Purge Transaction Matching PowerShell script

$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$epmusername="$($inputproperties.epmusername)"
$epmpassword="$($inputproperties.epmpassword)"
$epmurl="$($inputproperties.epmurl)"
$objectstorageusername="$($inputproperties.objectstorageusername)"
$objectstoragepassword="$($inputproperties.objectstoragepassword)"
$objectstorageurl="$($inputproperties.objectstorageurl)"
$matchtype="$($inputproperties.matchtype)"
$age="$($inputproperties.age)"
$filteroperator="$($inputproperties.filteroperator)"
$filtervalues="$($inputproperties.filtervalues)"
$proxyserverusername="$($inputproperties.proxyserverusername)"
$proxyserverpassword="$($inputproperties.proxyserverpassword)"
$proxyserverdomain="$($inputproperties.proxyserverdomain)"

echo "Running processes to archive and purge transaction matching
transactions ..."
if (${proxyserverusername}) {
    echo "Running epmautomate login ${epmusername} ${epmpassword} ${epmurl} $
{proxyserverusername} ${proxyserverpassword} ${proxyserverdomain}"
    epmautomate login ${epmusername} ${epmpassword} ${epmurl} $
{proxyserverusername} ${proxyserverpassword} ${proxyserverdomain}
} else {
    echo "Running epmautomate login ${epmusername} ${epmpassword} ${epmurl}"
    epmautomate login ${epmusername} ${epmpassword} ${epmurl}
}
echo "Running epmautomate archiveTmTransactions \"${matchtype}\" ${age}
filterOperator=${filteroperator} ${filtervalues}"
```

```
epmautomate archiveTmTransactions "${matchtype}" "${age}" "filterOperator=$
{filteroperator}" "${filtervalues}" > ./outfile.txt.tmp

$jobIdLine=Select-String -Path "outfile.txt.tmp" -Pattern "Job
ID"; $jobIdLine=($jobIdLine -split ":")[-2]; $jobid=($jobIdLine -split " ")
[1];
$logfile="Archive_Transactions_${matchtype}_${jobid}.log"
$transactionsfile="Archive_Transactions_${matchtype}_${jobid}.zip"
epmautomate listfiles > ./files.txt.tmp
$transactionslogfound=Select-String -Path "./files.txt.tmp" -Pattern "$
{logfile}"
$transactionsfilefound=Select-String -Path "./files.txt.tmp" -Pattern "$
{transactionsfile}"
rm ./outfile.txt.tmp
rm ./files.txt.tmp

if (${transactionslogfound}) {
    echo "Running epmautomate downloadfile \"${logfile}\""
    epmautomate downloadfile "${logfile}"
    if (${transactionsfilefound}) {
        echo "Running epmautomate downloadfile ${transactionsfile}"
        epmautomate downloadfile "${transactionsfile}"
        if ($?) {
            echo "Running epmautomate copyToObjectStorage $
{transactionsfile} ${objectstorageusername} ${objectstoragepassword} $
{objectstorageurl}"
            epmautomate copyToObjectStorage "${transactionsfile}" "$
{objectstorageusername}" "${objectstoragepassword}" "${objectstorageurl}"
            if ($?) {
                echo "Running epmautomate purgeArchivedTMTransactions JOBID=$
{jobid}"
                epmautomate purgeArchivedTMTransactions "JobID=${jobid}"
            }
        else {
                echo "EPMAutomate copyToObjectStorage failed. Purging of
archived matched transactions will not be attempted."
            }
    }
    else {
            echo "Download of transactions file ${transactionsfile} failed.
Copy to object storage, and purging of archived matched transactions will not
be attempted."
    }
    }
    else {
        echo "No matched transactions found. Archive file download, copy to
object storage, and purging of archived matched transactions will not be
attempted."
    }
}
else {
    echo "Matchtype ID \"${matchtype}\" not found. Archive file download,
copy to object storage, and purging of archived matched transactions will not
be attempted."
}
```

```
echo "Running epmautomate logout"
epmautomate logout
echo "Script processing completed."
```

Linux/UNIX Script

Create `transaction_match.sh` by copying the following script. Store it in the folder where `input.properties` is stored.

```
#!/bin/sh

. ./input.properties
export JAVA_HOME=${javahome}

echo "Running processes to archive and purge transaction matching
transactions..."
if [[ "${proxyserverusername}" != "" ]]
then
    echo "Running epmautomate login ${epmusername} ${epmpassword} ${epmurl}
ProxyServerUserName=${proxyserverusername} ProxyServerPassword=$
{proxyserverpassword} ProxyServerDomain=${proxyserverdomain}"
    ${epmautomatescript} login "${epmusername}" "${epmpassword}" "${epmurl}"
"ProxyServerUserName=${proxyserverusername}" "ProxyServerPassword=$
{proxyserverpassword}" "ProxyServerDomain=${proxyserverdomain}"
else
    echo "Running epmautomate login ${epmusername} ${epmpassword} ${epmurl}"
    ${epmautomatescript} login "${epmusername}" "${epmpassword}" "${epmurl}"
fi
echo "Running epmautomate archiveTmTransactions \"${matchtype}\" ${age}
filterOperator=${filteroperator} ${filtervalues}"
${epmautomatescript} archiveTmTransactions "${matchtype}" "${age}"
"filterOperator=${filteroperator}" "${filtervalues}" > ./output.txt.tmp

jobid=$(grep "Job ID" ./output.txt.tmp | cut -d':' -f3 | cut -d' ' -f2)
logfile="Archive_Transactions_${matchtype}_${jobid}.log"
transactionsfile="Archive_Transactions_${matchtype}_${jobid}.zip"
${epmautomatescript} listfiles > ./files.txt.tmp
transactionslogfound=$(grep "${logfile}" ./files.txt.tmp | wc -l)
transactionsfilefound=$(grep "${transactionsfile}" ./files.txt.tmp | wc -l)
rm ./files.txt.tmp
rm ./output.txt.tmp

if [ ${transactionslogfound} -eq 0 ]
then
    echo "Matchtype ID \"${matchtype}\" not found. Archive file download,
copy to object storage, and purging of archived matched transactions will not
be attempted."
else
    echo "Running epmautomate downloadfile \"${logfile}\""
    ${epmautomatescript} downloadfile "${logfile}"
    if [ ${transactionsfilefound} -eq 0 ]
    then
        echo "No matched transactions found. Archive file download, copy to
object storage, and purging of archived matched transactions will not be
attempted."
    else
```

```
        echo "Running epmautomate downloadfile ${transactionsfile}"
        ${epmautomatescript} downloadfile "${transactionsfile}"
        if [ $? -eq 0 ]
        then
            echo "Running epmautomate copyToObjectStorage $
{transactionsfile} ${objectstorageusername} ${objectstoragepassword} $
{objectstorageurl}"
            ${epmautomatescript} copyToObjectStorage "${transactionsfile}" "$
{objectstorageusername}" "${objectstoragepassword}" "${objectstorageurl}"
            if [ $? -eq 0 ]
            then
                echo "Running epmautomate purgeArchivedTMTransactions JOBID=$
{jobid}"
                ${epmautomatescript} purgeArchivedTMTransactions "JobID=$
{jobid}"
            else
                echo "EPMAutomate copyToObjectStorage failed. Purging of
archived matched transactions will not be attempted."
            fi
        else
            echo "Download of transactions file ${transactionsfile} failed.
Copy to object storage, and purging of archived matched transactions will not
be attempted."
        fi
    fi
fi

echo "Running epmautomate logout"
${epmautomatescript} logout
echo "Script processing completed."
```

Sample Scenarios for Profitability and Cost Management

These scenarios explore the use of EPM Automate commands to perform some common Profitability and Cost Management tasks.

Related Topics

- Import Metadata into Application
  Use these scripts to upload a metadata file and to import dimension metadata from it into an Profitability and Cost Management application.
- Import Data and Run Program Rules
  Use these scripts to upload data files and to import data from the uploaded files into a Profitability and Cost Management business process.

Import Metadata into Application

Use these scripts to upload a metadata file and to import dimension metadata from it into an Profitability and Cost Management application.

These scripts perform the following operations:

- Signs in to the environment.
- Uploads a metadata file.

- Imports metadata from the uploaded file into an application.
- Enables the application.
- Signs out.

Windows Script

Create `importMetadata.ps1` by copying this script.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$applicationName="$($inputproperties.applicationName)"
$dataFileName="$($inputproperties.dataFileName)"
$dataFileNameDestination="$($inputproperties.dataFileNameDestination)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile "${dataFileName}" ${dataFileNameDestination}
epmautomate loaddimdata ${applicationName} dataFileName=${dataFileName}
epmautomate enableapp ${applicationName}
epmautomate logout
```

Linux/UNIX Script

Create `importMetadata.sh` by copying this script.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${dataFileName}" "${dataFileNameDestination}"
${epmautomatescript} loaddimdata "${applicationName}" "dataFileName=$
{dataFileName}"
${epmautomatescript} enableapp "${applicationName}"
${epmautomatescript} logout
```

Creating the input.properties File

To run the `importMetadata` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `importMetadata.ps1` or `importMetadata.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
applicationName=APPLICATION_NAME
dataFileName=DATA_FILE.txt
dataFileNameDestination=profitinbox
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
applicationName=APPLICATION_NAME
dataFileName=DATA_FILE.txt
dataFileNameDestination=profitinbox
```

Table 3-25 input.properties Parameters

Parameter	Description
javahome	JAVA_HOME location. For Linux/UNIX only.
epmautomatescript	Absolute path of EPM Automate executable (epmautomate.sh). For Linux/UNIX only.
username	User name of a Service Administrator, who also has the Identity Domain Administrator role.
password	Password of the Service Administrator or the name and location of the encrypted password file.
serviceURL	URL of the environment from which you want to generate the snapshot.
applicationName	Name of the Profitability and Cost Management into which data is to be loaded.
dataFileName	Name of the file containing the metadata to be imported.
dataFileNameDestination	Upload location for the metadata file.

Running the Scripts

1. Create `importMetadata.ps1` or `importMetadata.sh` by copying the script from a preceding section.
2. Create the `input.properties` file and save it in the directory where the `importMetadata` script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.
3. Launch the script.
   - **Windows PowerShell:** run `importMetadata.ps1`.
   - **Linux/UNIX:** run `./importMetadata.sh`.

Import Data and Run Program Rules

Use these scripts to upload data files and to import data from the uploaded files into a Profitability and Cost Management business process.

These scripts complete the following steps:

- Sign in to the environment.
- Uploads a data file containing the data to load.

- Uploads rule file containing data rules.
- Loads data from the data file into application to overwrite existing values.
- Runs all rules in the rule file.
- Signs out.

Windows Script

Create a file named `importData.ps1` by copying the following script. Store it in a local directory.

```
$inputproperties = ConvertFrom-StringData(Get-Content ./input.properties -raw)
$username="$($inputproperties.username)"
$passwordfile="$($inputproperties.passwordfile)"
$serviceURL="$($inputproperties.serviceURL)"
$applicationName="$($inputproperties.applicationName)"
$dataFileName="$($inputproperties.dataFileName)"
$rulesFileName="$($inputproperties.rulesFileName)"
$fileDestination="$($inputproperties.fileDestination)"
$clearAllDataFlag="$($inputproperties.clearAllDataFlag)"
$dataLoadValue="$($inputproperties.dataLoadValue)"

epmautomate login ${username} ${passwordfile} ${serviceURL}
epmautomate uploadfile "${dataFileName}" ${fileDestination}
epmautomate uploadfile "${rulesFileName}" ${fileDestination}
epmautomate loaddata ${applicationName} clearAllDataFlag=${clearAllDataFlag}
dataLoadValue=${dataLoadValue} rulesFileName="${rulesFileName}"
dataFileName="${dataFileName}"
epmautomate logout
```

Linux/UNIX Script

Create a file named `importData.sh` by copying the following script. Store it in a local directory.

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${dataFileName}" "${fileDestination}"
${epmautomatescript} uploadfile "${rulesFileName}" "${fileDestination}"
${epmautomatescript} loaddata "${applicationName}" "clearAllDataFlag=$
{clearAllDataFlag}" "dataLoadValue=${dataLoadValue}" rulesFileName="$
{rulesFileName}" dataFileName="${dataFileName}"
${epmautomatescript} logout
```

Creating the input.properties File

To run the `importData` scripts, create the `input.properties` file and update it with information for your environment. Save the file in the directory where `importData.ps1` or `importData.sh` is stored.

Windows

```
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
```

```
applicationName=APPLICATION_NAME
dataFileName=DATA_FILE.txt
rulesFileName=RULE_FILE.txt
fileDestination=profitinbox
clearAllDataFlag=true
dataLoadValue=OVERWRITE_EXISTING_VALUES
```

Linux/UNIX

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
passwordfile=examplePassword.epw
serviceURL=exampleURL
applicationName=APPLICATION_NAME
dataFileName=DATA_FILE.txt
rulesFileName=RULE_FILE.txt
fileDestination=profitinbox
clearAllDataFlag=true
dataLoadValue=OVERWRITE_EXISTING_VALUES
```

Table 3-26 input.properties Parameters

Parameter	Description
javahome	JAVA_HOME location. For Linux/UNIX only.
epmautomatescript	Absolute path of EPM Automate executable (epmautomate.sh). For Linux/UNIX only.
username	User name of a Service Administrator, who also has the Identity Domain Administrator role.
password	Password of the Service Administrator or the name and location of the encrypted password file.
serviceURL	URL of the environment from which you want to generate the snapshot.
applicationName	Name of the Profitability and Cost Management into which data is to be loaded.
dataFileName	Name of the file containing the data to be imported.
rulesFileName	Name of the file containing the rules to be run on the imported data.
fileDestination	Location to which the data and rules files are to be uploaded.
clearAllDataFlag	Specifies whether to clear existing data in the application cube. Set to false if you do not want to clear existing data.
dataLoadValue	specifies how to handle existing data. Specify ADD_TO_EXISTING if you want to retaining existing data in the cube.

Running the Scripts

1. Create importData.ps1 or importData.sh by copying the script from a preceding section.
2. Create the input.properties file and save it in the directory where the importData script is located. Contents of this file differs depending on your operating system. See Creating the input.properties File.
   Make sure that you have write privileges in this directory. For Windows, you may need to start PowerShell using the **Run as Administrator** option to be able to run the script.

3. Launch the script.
   - **Windows PowerShell:** run `importData.ps1`.
   - **Linux/UNIX:** run `./importData.sh`.

Sample Scenarios for Oracle Enterprise Data Management Cloud

These sample scenarios explore using EPM Automate commands to synchronize application dimensions between Oracle Enterprise Data Management Cloud and Oracle Enterprise Performance Management Cloud.

Related Topics

- Synchronizing Oracle Enterprise Data Management Cloud Dimensions and Mappings with EPM Cloud Applications
  This sample scenario explores synchronizing a dimension between an Oracle Enterprise Data Management Cloud application and an Oracle Enterprise Performance Management Cloud application.
- Synchronizing EPM Cloud Dimensions with Oracle Enterprise Data Management Cloud Applications
  This sample scenario explores synchronizing a dimension between an Oracle Enterprise Performance Management Cloud application and an Oracle Enterprise Data Management Cloud application.

Synchronizing Oracle Enterprise Data Management Cloud Dimensions and Mappings with EPM Cloud Applications

This sample scenario explores synchronizing a dimension between an Oracle Enterprise Data Management Cloud application and an Oracle Enterprise Performance Management Cloud application.

You use the scripts in this section to complete these tasks:

- Export a dimension from an Oracle Enterprise Data Management Cloud application
- Export mappings from an Oracle Enterprise Data Management Cloud application dimension
- Copy the export files to an EPM Cloud environment
- Import dimension metadata and mappings into the EPM Cloud application

To synchronize a dimension and mappings between an Oracle Enterprise Data Management Cloud application and an EPM Cloud application:

1. Create a script file by copying the following script:

```
rem Integration example to sync application dimensions between EDM and EPM
Cloud
rem Windows script for demonstration purposes only; do not use in
production environments

set EDMUSER=userid
set EDMSVR=https://hostname
set EDMPWDFILE=example_EDM
```

```
set EDMAPP=appname
set EDMDIM=dimname
set EDMLOC=location

set EPMUSER=userid
set EPMSVR=https://hostname
set EPMIMPJOB=importjobname
set PWDFILE=C:\Oracle\EPM.epw
set DIMFILE=dimension.csv
set MAPFILE=mapping.csv

rem Synchronizing EDM ---> EPM
rem Export Dimension and Mappings from EDM

call epmautomate login %EDMUSER% %EDMPWDFILE% %EDMSVR%
call epmautomate exportdimension %EDMAPP% %EDMDIM% %DIMFILE%
call epmautomate exportdimensionmapping %EDMAPP% %EDMDIM% %EDMLOC%
%MAPFILE%
call epmautomate logout

rem Log into the EPM Cloud environment
call epmautomate login %EPMUSER% %PWDFILE% %EPMSVR%

rem Copy exported files from EDM environment to EPM and import metadata
and mappings
call epmautomate copyfilefrominstance %DIMFILE% %EDMUSER% %EDMPWDFILE%
%EDMSVR% inbox/%DIMFILE%
call epmautomate importmetadata %EPMIMPJOB%

call epmautomate copyfilefrominstance %MAPFILE% %EDMUSER% %EDMPWDFILE%
%EDMSVR% inbox/%MAPFILE%
call epmautomate importmapping %EDMDIM% %MAPFILE% REPLACE FALSE %EDMLOC%

call epmautomate logout
```

2. Modify the script file and set the required parameter values. See Parameters for Script Execution for explanation and example of the parameters.
3. Run the script manually or schedule it to run as needed. See Automating Script Execution.

Parameters for Script Execution

The script files in this section requires you to specify some of the parameter values explained in the following table. Not all these parameters are used in all the scripts.

Table 3-27 Parameter Values for Script Files

Parameter	Description
EDMUSER	User login ID of a Oracle Enterprise Data Management Cloud Service Administrator. **Example:** EDMUSER=jdoe@example.com
EDMSVR	URL of the Oracle Enterprise Data Management Cloud environment. **Example:** EDMSVR=https:// example.oraclecloud.com

Table 3-27 (Cont.) Parameter Values for Script Files

Parameter	Description
EDMPWDFILE	Name and location of the encrypted password file (EPW) of the Oracle Enterprise Data Management Cloud Service Administrator. **Example:** EDMPWDFILE=edm_jdoe.epw
EDMAPP	Name of an Oracle Enterprise Data Management Cloud application dimension. **Example:** EDMAPP=USOperations
EDMDIM	Name of the dimension to export or import. **Example:** EDMDIM=entity
EDMLOC	Name of the location to export. **Example:** EDMLOC=Loc1
EPMUSER	Login name of an EPM Cloud Service Administrator. **Example:** EPMUSER=john.doe@example.com
EPMSVR	URL of the EPM Cloud environment. **Example:** EPMSVR=https://example.oraclecloud.com
EPMIMPJOB	Name of an existing import job of type import metadata in the EPM Cloud environment. **Example:** EPMIMPJOB=imp_DIMMetadata
EPMEXPJOB	Name of an existing job of type export metadata in theEPM Cloud environment. **Example:**EPMEXPJOB=Exp_DIMMetadata
PWDFILE	Name and location of the encrypted password file (EPW) for EPM Cloud Service Administrator. See the encrypt command. **Example:** PWDFILE=pwd_jdoe.epw
DIMFILE	Name of the file to hold exported dimension data. **Example:** DIMFILE=entity_file.CSV
MAPFILE	Name of the file to hold exported mapping data. **Example:** MAPFILE=map_file.CSV

Synchronizing EPM Cloud Dimensions with Oracle Enterprise Data Management Cloud Applications

This sample scenario explores synchronizing a dimension between an Oracle Enterprise Performance Management Cloud application and an Oracle Enterprise Data Management Cloud application.

You use the scripts in this section to complete these tasks:

- Export metadata (dimensions) from an EPM Cloud application
- Copy the export files containing dimension data to an Oracle Enterprise Data Management Cloud environment
- Import dimension metadata into the Oracle Enterprise Data Management Cloud application

To synchronize a dimension between an EPM Cloud application and an Oracle Enterprise Data Management Cloud application:

1. Create a Windows script file by copying the following script:

```
rem Integration example to sync an application dimension between EPM Cloud
and EDM
rem Windows script for demonstration purposes only; do not use in
production environments

set EDMUSER=userid
set EDMSVR=https://hostname
set EDMPWDFILE=example_EDM.epw
set EDMAPP=appname
set EDMDIM=dimname

set EPMUSER=userid
set EPMSVR=https://hostname
set PWDFILE=example_epm.epw
set EPMEXPJOB=exportjobname

rem Synchronizing EPM ---> EDM

rem Export Metadata from EPM
call epmautomate login %EPMUSER% %PWDFILE% %EPMSVR%
call epmautomate exportmetadata %EPMEXPJOB%
call epmautomate logout

rem Import Dimension to EDM
rem Log into the EDM environment
call epmautomate login %EDMUSER% %EDMPWDFILE% %EDMSVR%
rem Copy exported metadata file into the EDM environment
call epmautomate copyfilefrominstance %EPMEXPJOB%.zip %EPMUSER% %PWDFILE%
%EPMSVR% %EPMEXPJOB%.zip
call epmautomate importdimension %EDMAPP% %EDMDIM% ReplaceNodes
%EPMEXPJOB%.zip
call epmautomate logout
```

   Modify the script file and set the required parameter values. See Parameters for Script Execution for explanation and example of the parameters.

2. Run the script manually or schedule it to run as needed. See Automating Script Execution.

Automating Script Execution

A Service Administrator schedules scripts in Windows Task Scheduler or uses a cron job to automate activities using EPM Automate.

To schedule EPM Automate script execution using Windows Task Scheduler:

1. Click **Start**, then **Control Panel**, and then **Administrative Tools**.
2. Open **Task Scheduler**.
3. Select **Action**, and then **Create Basic Task**.
4. Enter a task name and an optional description, and then click **Next**.
5. In **Task Trigger**, select a schedule for running the script, and then click **Next**.
6. In the next screen, specify other schedule parameters, and then click **Next**.

7. In Action, ensure that **Start a program** is selected.
8. In **Start a Program**, complete these steps:
    a. In **Program/script**, browse and select the script that you want to schedule.
    b. In **Add arguments (optional)**, enter the password of the Service Administrator identified by the `SET user` script parameter.
    c. In **Start in (optional)**, enter the location where EPM Automate is installed; generally, `C:/Oracle/EPMAutomate/bin`.
    d. Click **Next**.
9. In **Summary**, select **Open the Properties dialog for this task when I click Finish**, and then click **Finish**.
10. In **General**, select these security options, and then click **OK**.
    - **Run whether user is logged in or not**
    - **Run with highest privileges**

Monitoring EPM Automate Activities

To help you identify the status of the operation that you initialized, EPM Automate displays status codes in the console from which you run it.

See Exit Codes.

Use the Job Console to monitor the jobs that you execute using EPM Automate. See Managing Jobs in *Administering Planning* for details.

4

Running Commands without Installing EPM Automate

Using Groovy, you can run select commands directly in Oracle Enterprise Performance Management Cloud. You do not require an EPM Automate installation to run server-side commands.

> **Note:**
>
> In this scenario, Groovy script is written to be executed directly in EPM Cloud, not on any client machine.

In this Chapter:

Environments that Support Server-side Command Execution

Groovy scripting support for server-side execution of EPM Automate commands is available in the following environments only:

- Planning and Planning Modules business processes deployed in EPM Enterprise Cloud Service environments.
- Enterprise Planning and Budgeting Cloud
- Planning and Budgeting Cloud with Plus One option
- FreeForm
- Enterprise Profitability and Cost Management
- Financial Consolidation and Close in EPM Enterprise Cloud Service environments.
- Tax Reporting in EPM Enterprise Cloud Service environments.
- Strategic Workforce Planning
- Sales Planning

Groovy scripts incorporating EPM Automate commands can be written and executed in the preceding Oracle Enterprise Performance Management Cloud environments only. Scripts written in such environments can, however, issue EPM Automate commands on any EPM Cloud environment. For example, you can create the script in a Planning EPM Enterprise

Cloud Service environment and have it issue commands on a Narrative Reporting environment that does not support groovy scripting.

Information Sources

See these sources in *Designing with Calculation Manager for Oracle Enterprise Performance Management Cloud* for detailed information:

- About Groovy Business Rules
- Creating a Groovy Business Rule

Supported Commands

Except for the following, all EPM Automate commands can be run through Groovy:

- downloadFile
- upgrade
- uploadFile

The following commands cannot be executed on the environment running the Groovy script:

- recreate
- replay
- resetService
- restructureCube

> **Note:**
>
> - On running the encrypt command, the encrypted password file is created on the server; it is purged after seven days if not used.
> - For the feedback command to work, all files and screenshots used as attachment should be available in the default upload location. See Default Upload Location. This is the location where files are stored if you do not specify a location in the uploadFile command.

Methods to be Used for Running EPM Automate Using Server-Side Groovy

- `getEPMAutomate ()` This static method provides an instance of `EpmAutomate` class, which can then be used to invoke EPM Automate commands.
- `execute ()` This method of `EpmAutomate` class is used to execute an EPM Automate command. Pass the EPM Automate command name as the first parameter and command options as subsequent parameters. This method returns an instance of `EpmAutomateStatus` class.

- `getStatus ()` This method of `EPMAutomateStatus` class returns the execution status returned by the command. Return value 0 indicates success while a non-zero value means command failure.
- `getOutput ()` This method of `EPMAutomateStatus` class returns the string output of the command. For example, you can use this method to return the output of the `getApplicationAdminMode` and `getDailyMaintenanceStartTime` command. If the return status of the command is non-zero, this method returns the error message.
- `getItemsList ()` This method of `EPMAutomateStatus` class returns the list output of the command. For example, you can use this method to return the output of the `getSubstVar`, `listBackups`, and `listFiles` commands.

Cloning an Environment Using a Server-Side Groovy Script

You can include EPM Automate commands in server-side Groovy scripts to clone environments for disaster recovery purposes. This allows setting up disaster recovery without any on-premises footprint.

If passwords contain special characters, see Handling Special Characters. Also, be sure to replace these parameter values to suit your environments:

Table 4-1 Parameters to Change

Parameter	Description
`password`	The password of the Service Administrator who is performing the clone operation in the source environment.
`targetpassword`	The password of the Service Administrator who is performing the clone operation in the target environment.
`username`	The user ID of the Service Administrator in the source environment.
`targetusername`	The user ID of the Service Administrator in the target environment. This user must also be assigned to the Identity Domain Administrator role in the target environment.
`email_id`	The email address to which you plan to send information about the cloning process.

Script for Encrypting Password

```
EpmAutomate automate = getEpmAutomate()

//Encrypt the password of a Service Administrator in the source environment
EpmAutomateStatus encryptstatus1 = automate.execute('encrypt',
'password','encryptionKey','sourcePassword.epw')
if(encryptstatus1.getStatus() != 0)
throwVetoException(encryptstatus1.getOutput())
println(encryptstatus1.getOutput())

//Encrypt the password of a Service Administrator in the target environment
//This user must also have the Identity Domain Administrator
//role in the target environment

EpmAutomateStatus encryptstatus2= automate.execute('encrypt',
'targetpassword',
'encryptionKey', 'targetPassword.epw')
```

```
if(encryptstatus2.getStatus() != 0)
throwVetoException(encryptstatus2.getOutput())
println(encryptstatus2.getOutput())
```

Script for Cloning the Environment

This script uses the encrypted password files created using the preceding script.

```
EpmAutomate automate = getEpmAutomate()

//Log into the target environment
EpmAutomateStatus loginstatus = automate.execute('login',
'username','targetPassword.epw' , 'targeturl')
if(loginstatus.getStatus() != 0) throwVetoException(loginstatus.getOutput())
println(loginstatus.getOutput())

//Recreate the target environment
EpmAutomateStatus recreatestatus = automate.execute('recreate' , '-f' )
if(recreatestatus.getStatus() != 0)
throwVetoException(recreatestatus.getOutput())
println(recreatestatus.getOutput())

//Copy Artifact Snapshot from the source environment
//to the target environment
EpmAutomateStatus copystatus = automate.execute('copysnapshotfrominstance',
'Artifact Snapshot', 'sourceusername', 'sourcePassword.epw','source url')
if(copystatus.getStatus() != 0) throwVetoException(copystatus.getOutput())
println(copystatus.getOutput())

//import Artifact Snapshot into the target environment
EpmAutomateStatus importstatus = automate.execute('importsnapshot', 'Artifact
Snapshot')
println(importstatus.getOutput())

//Send an email to a designated user with the status of the
//snapshot import process
EpmAutomateStatus emailstatus = automate.execute('sendmail',
'email_id' ,'Status of DR' , 'BODY='+ importstatus.getOutput())
println(emailstatus.getOutput())

//Sign out of the target environment
EpmAutomateStatus logoutstatus = automate.execute('logout')
println(logoutstatus.getOutput())
```

Emailing the Activity Report Using a Server-side Groovy Script

This script can be used to email the Activity Report to a list of recipients. This script can then be scheduled to run daily to get the daily Activity Report. This script performs the following functions:

- Encrypts the password of the Service Administrator who executes the Groovy script.
- Signs into the Oracle Enterprise Performance Management Cloud environment using the encrypted password.

- Emails the Activity Report available in the environment to a list of recipients, generally, Service Administrators
- Signs out of the environment.

If passwords contain special characters, see Handling Special Characters. Also, be sure to replace these parameter values to suit your environments:

Table 4-2 Parameters to Change

Parameter	Description
user	The user ID of a Service Administrator to sign into the environment.
password	The password of the Service Administrator.
url	URL of the EPM Cloud environment from which the Activity Report is to be emailed.
emailaddresses	A semicolon separated list of email addresses to which the Activity Report is to be sent.

For detailed information on using Groovy rules, see Using Groovy Rules in *Administering Planning*.

```
/*RTPS: {user} {password} {url} {emailaddresses}*/
import java.text.SimpleDateFormat

String user = 'service_administrator'
String password = 'examplePWD'
String url = 'example_EPM_URL'
String emailaddresses = 'service_administrator@oracle.com'

EpmAutomate automate = getEpmAutomate()

def LogMessage(String message) {
    def date = new Date()
    def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss")
    println('[' + sdf.format(date) + '][GROOVY] ' + message);
}

def LogOperationStatus(EpmAutomateStatus opstatus) {
    def returncode = opstatus.getStatus()
    LogMessage(opstatus.getOutput())
    LogMessage('return code: ' + returncode)
}

LogMessage('Starting mail activity report processing')

// encrypt
LogMessage("Operation: encrypt " + password + " oracleKey password.epw")
EpmAutomateStatus status =
automate.execute('encrypt',password,"oracleKey","password.epw")
LogOperationStatus(status)

// login
LogMessage("Operation: login " + user + " password.epw " + url)
status = automate.execute('login',user,"password.epw",url)
LogOperationStatus(status)
```

```
// listfiles
LogMessage('Operation: listfiles')
status = automate.execute('listfiles')
LogOperationStatus(status)

String filelist = status.getItemsList()
String[] str = filelist.split(',');
String reportfile = ''

for( String svalues : str ) {
    String[] ftr = svalues.split('/')
    for( String fvalues : ftr ) {
        if (fvalues.startsWith('2') && fvalues.endsWith('html')) {
            reportfile = fvalues
        }
    }
}

def reportdir = reportfile.tokenize(".")[0]
String reportpath = 'apr/' + reportdir + '/' + reportfile

// sendMail
LogMessage('Operation: sendMail ' + emailaddresses + ' Daily Activity Report
Body=Daily Activity Report Attachments=' + reportpath)
status = automate.execute('sendmail',emailaddresses,'Daily Activity
Report','Body=Daily Activity Report',"Attachments=${reportpath}")
LogOperationStatus(status)

// logout
LogMessage('Operation: logout')
status = automate.execute('logout')
LogOperationStatus(status)
```

5

Replicating an EPM Cloud Environment

These steps are involved in configuring a secondary Oracle Enterprise Performance Management Cloud environment to ensure availability of service if the primary Oracle data center becomes unavailable due to unforeseen circumstances.

> **Note:**
>
> The procedures discussed in this appendix are not applicable to Narrative Reporting.

- Setting up daily artifact replication
- Setting up On-Demand Replications
- Configuring the Secondary Environment

Setting up Daily Replication

To replicate an environment, you use EPM Automate to copy the artifact snapshot created during the daily maintenance from the primary environment to the secondary environment.

Oracle performs routine maintenance on each environment on a daily basis. During this service maintenance, Oracle creates a maintenance snapshot by backing up the contents of the environment (existing data and artifacts, including user and role assignments from the identity domain).

To setup daily service replication:

1. Create a script file that contains the following EPM Automate commands. This script replicates the application snapshot from the primary environment to the secondary environment.

   > **Note:**
   >
   > Be sure to change the user name, password file, identity domain names, and service URLs. For information on creating an encrypted password file, see the encrypt command.

   ```
   REM Sign in to the secondary instance
   epmautomate login serviceAdmin secondaryPassword.epw secondary_URL
   secondaryDomain
   REM Delete the existing artifact snapshot
   epmautomate deletefile "Artifact Snapshot"
   REM Copy the snapshot from the primary instance
   epmautomate copysnapshotfrominstance "Artifact Snapshot"
   primaryPassword.epw primary_URL primaryDomain
   ```

```
REM Sign out of the secondary instance
epmautomate logout
```

2. Using a scheduler; for example, Windows Task Scheduler, schedule the execution of the script file so that it runs two hours from the beginning of the maintenance window.
3. Set identical Maintenance Window start time on both the primary and secondary environments. See Setting Service Maintenance Time in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators* for more information.

Setting up On-Demand Replications

To reduce the RPO, you can create on-demand snapshots of the primary environment and then copy them to the secondary environment.

For example, you can create and schedule an EPM Automate script that runs every six hours between the daily replications to reduce the RPO from 24 to six hours.

> **Note:**
>
> During on-demand snapshot creation, the primary environment is placed in read only mode for a few minutes.

To setup on-demand replication:

1. Create a script file that contains the following EPM Automate commands. This script replicates the application snapshot from the primary environment to the secondary environment.

   > **Note:**
   >
   > Be sure to change the user name, password file, identity domain names, and service URLs. For information on creating an encrypted password file, see the encrypt command.

```
REM Sign in to the primary instance
epmautomate login serviceAdmin primaryPassword.epw primary_URL
primaryDomain
REM Create a snapshot and then sign out
epmautomate exportsnapshot "Artifact Snapshot"
epmautomate logout
REM Sign in to the secondary instance
epmautomate login serviceAdmin secondaryPassword.epw secondary_URL
secondaryDomain
REM Copy the snapshot from the primary instance
epmautomate copysnapshotfrominstance "Artifact Snapshot"
primaryPassword.epw primary_URL primaryDomain
REM Sign out of the secondary instance
epmautomate logout
```

2. Using a scheduler; for example, Windows Task Scheduler, schedule the execution of the script file to run as needed to meet the desired RPO.

Configuring the Secondary Environment

You configure the secondary environment to activate it.

Complete this procedure only if you need to activate the secondary environment when the primary environment is unavailable for an extended period. Before configuring the secondary environment, refer to these topics in *Getting Started with Oracle Enterprise Performance Management Cloud for Administrators*:

- Migration Paths for Legacy EPM Cloud Snapshots
- Migration Paths for EPM Standard Cloud Service and EPM Enterprise Cloud Service Snapshots

To configure the secondary environment:

1. Start an EPM Automate session and complete these activities.
   - Sign in to the secondary environment using an account that has both the Service Administrator and the Identity Domain Administrator roles. Be sure to specify the appropriate user name, password, domain name, and service URL.
   - Re-create the secondary environment.
     - If the primary environment is a Planning, Tax Reporting, Financial Consolidation and Close or Enterprise Profitability and Cost Management environment, use:
       `epmautomate recreate -f`
     - If the primary environment is not a Planning, Tax Reporting, Financial Consolidation and Close or Enterprise Profitability and Cost Management environment, use:
       `epmautomate recreate -f TempServiceType=`*`PRIMARY_APPLICATION_TYPE`*, where *`PRIMARY_APPLICATION_TYPE`* is `ARCS`, `EDMCS`, `PCMCS`, or `EPRCS`.
   - Import application and identity domain artifacts from the snapshot.
   - Sign out.

   You can complete the preceding activities by running the following commands. See these topics:
   - login command
   - recreate command
   - importSnapshot command

   ```
   epmautomate login serviceAdmin secondaryPassword.epw secondary_URL
   epmautomate recreate -f
   epmautomate importsnapshot "Artifact Snapshot" importUsers=true
   epmautomate logout
   ```

2. Sign into the secondary environment and verify that all data is available.
3. Send the URL of the secondary environment to all users.

A

Preparing to Run the simulateConcurrentUsage Command

The simulateConcurrentUsage command supports these operations to simulate load on an environment:

- Open forms
- Save forms
- Run business rules
- Run data rules
- Open ad hoc grids
- Execute Management Report Books
- Execute Management Report Reports

Steps Involved

1. Create the `requirement.csv` file. See Creating the requirement.csv File
2. Create input files specifying the details of the operations included in `requirement.csv`. See:
   - Open Form Input File
   - Save Form Input File
   - Run Business Rule Input File
   - Run Data Rule Input File
   - Ad Hoc Grid Input File
   - Execute Book Input File
   - Execute Report Input File
3. Create `UserVarMemberMapping.csv` containing the user variable details if user variable member mapping needs to be set. See Creating the UserVarMemberMapping.csv File
4. Optionally, export Oracle Smart View for Office options into `options.xml`, if Smart View options need to be used. See Creating the options.xml File for details.
5. Optionally, create the `users.csv` file to run simulations using existing users. See Creating the users.csv File for details.
6. Create a ZIP file containing the files that you created in the preceding steps and upload it to the environment. See Creating and Uploading the Input ZIP File to the Environment
7. Run the simulateConcurrentUsage command using the uploaded ZIP file.

Creating the requirement.csv File

Begin by creating the `requirement.csv` file that lists the details of the use cases that you want to test. Each line of this CSV file identifies the type of operation to perform, artifact name,

number of concurrent users, input file specifying the details of the operation, and additional, if any, information related to the operation. For example, to open 2 forms, save 2 forms, and run 2 business rules, you specify 6 lines in input CSV file. The first line of `requirement.csv` must contain this information:

```
#Type of Operation,Artifact Name,Number of Users,Input File,Additional Info
```

Each of the subsequent lines of the file contains a single operation and its parameters. Some operations may not require all these parameter values. The expected file entries are explained in the following table.

> **Note:**
>
> All values are required unless otherwise indicated in the table.

Table A-1 requirement.csv Format

Field	Description
Type of operation	Must be one of the following: • `Open Form` • `Save Form` • `Run Business Rule` • `Run Data Rule` • `Ad Hoc Grid` • `Execute Report` • `Execute Book`
Artifact Name	This value depends on the type of operation: • Open Form: The name and location of the form to open. • Save Form: The name and location of the form to save. • Run Business Rule: The name of the business rule. • Data Rule: The name of the data rule. • Ad Hoc Grid: Not applicable (leave blank). • Execute Report: The name and location of the of the report. • Execute Book: The name and location of the book.
Number of Users	Number of users to simulate concurrent use.
Input File	Name to the CSV file that contains the POV values, run-time prompts, or other use case-specific values to use
Additional Info	Additional parameters required for the operation. Applicable to Ad Hoc Grid only. Leave it blank for other use cases.

Notes: Artifact names must match those in the application and must be in the same case.

Example of a `requirement.csv` file:

```
# Type of Operation,Artifact Name,Number of Users,Input File,Additional Info
Open Form, Library/Global Assumption/Revenue Forecast
Assumptions,10,openform_input.csv,
Save Form, Library/Global Assumption/ExchangeRates,5,saveform_input.csv,
Run Business Rule, Run_FinStatement - Copy Budget to Prior Year
Budget,4,runbusinessrule_input.csv,
Run Data Rule, Delimited_file_DL,5,rundatarule_input.csv,
Ad Hoc Grid,,3,runadhocgrid_input.csv,cube=FinStmt
```

```
Execute Book,Review Books/Revenue Reports,10,book_input.csv
Execute Report,Review Reports/Executive Report,10,report_input.csv,
```

Creating the Input Files

Each use case identified in `requirement.csv` must have a matching input file that provides all the parameters required to execute it.

The input file, ideally, must contain one entry per the number of users specified for this use case in `requirement.csv`.

If the number of entries in the input file is less than the number of concurrent users for that use case in `requirement.csv`, EPM Automate repeats some entries from the input file to run the use cases until the operation is executed for the number of users identified in `requirement.csv`.

For example, if the use case entry in `requirement.csv` for the Run Business Rule operation is as follows:

```
Run Business Rule, Copy Budget,10,br_input_file.csv,
```

`br_input_file.csv` should also contain 10 entries. If `br_input_file.csv` contains only six entries, EPM Automate uses them for the first 6 users. For the next 4 users, EPM Automate reuses the first 4 entries in `br_input_file.csv`.

If the number of entries in the input file is more than the number of users specified for the use case, EPM Automate ignores the last extra entries in the input file.

Open Form Input File

This file, referenced in `requirement.csv` to support the opening of forms, includes POV entries in the format: `pov=[DIM 1:MEMBER 1],[DIM 2:MEMBER 2],[DIM 3:MEMBER 3],`, and so on.

In this discussion, DIM 1, DIM 2, and so on are the dimension names, and MEMBER 1, MEMBER 2, and so on are their dimension member values for the POV.

A sample input file:

```
pov=[Account:APL_RATE_AED],[Scenario:Budget],[Years:FY20]
pov=[Account:APL_RATE_AED],[Scenario:Budget],[Years:FY19]
pov=[Account:APL_RATE_AED],[Scenario:Budget],[Years:FY18]
pov=[Account:APL_RATE_AED],[Scenario:Budget],[Years:FY17]
pov=[Account:APL_RATE_AED],[Scenario:Budget],[Years:FY16]
```

> **Note:**
>
> You must also create `UserVarMemberMapping.csv` if the forms you specified in `requirement.csv` require user variables to be set. See Creating the UserVarMemberMapping.csv File.

Save Form Input File

This file, referenced in `requirement.csv` to support the saving of forms, includes POV, and cell input values in the following format. If business rules are associated with the form, this file also includes the runtime prompts for those business rules:

A sample input file:

```
pov=[DIM 1:MEMBER1],[DIM 2:MEMBER2],[DIM 3:MEMBER3],…;cells=[CELL COLUMN
HEADER 1 -> CELL COLUMN HEADER 2 -> CELL COLUMN HEADER 3 ->.. | CELL ROW
HEADER 1-> CELL ROW HEADER 2-> CELL ROW HEADER 3->..| CELL 1 DATA], [CELL
COLUMN HEADER 11 -> CELL COLUMN HEADER 22 -> CELL COLUMN HEADER 33 ->.. |
CELL ROW HEADER 11-> CELL ROW HEADER 22-> CELL ROW HEADER 33->..| CELL 2
DATA];rtp=[BUSINESS RULE NAME1[RTP1:VALUE1][RTP2:VALUE2]],[BUSINESS
RULE2[RTP3:VALUE3]]..
```

In this example:

- DIM indicates the name of a dimension and MEMBER indicates a dimension member value.
- CELL COLUMN HEADER identifies the name of the column header and CELL ROW HEADER identifies the name of a row header.
- BUSINESS RULE NAME indicates the name of the business rule, RTP indicates the runtime prompt name, and VALUE indicates its value. RTP is not required if no business rule is associated with the form or if you want to use the default runtime prompt value present in the form.

For example:

```
pov=[Version View:Working],[Sales Entity:International Sales];cells=[FY16-
>x---------x->Pct|P293:Maintenance->4120: Support|1];rtp=[Services Revenue -
Forecast[Department:000][Scenario:Plan]],[Allocate Plan
Targets[TargetVersion:Baseline]]
```

> **Note:**
>
> You must also create `UserVarMemberMapping.csv` if the forms you specify in `requirement.csv` require user variables to be set. See Creating the UserVarMemberMapping.csv File.

Smart Pushes, if any, associated with the form run automatically.

Run Business Rule Input File

This file, referenced in `requirement.csv` to support the running of business rules, includes runtime parameter values in the format: `rtp=[RTP1:Value1],[RTP2:Value2]`, and so on.

If no runtime parameter value is required for the business rule, then include `rtp=[]`. In this example, RTP1, RTP2, and so on identify the runtime prompt names and VALUE1, VALUE2 identify their values. Be sure to add as many runtime prompts as required.

A sample input file:

```
rtp=[Period:Q1],[Entity:USA]
rtp=[Period:Q2],[Entity:USA]
rtp=[Period:Q3],[Entity:USA]
rtp=[Period:Q4],[Entity:USA]
```

Run Data Rule Input File

This file, referenced in `requirement.csv` to support the running of data rules, should specify the start period, end period, import mode, export mode, and an optional import file name available in the environment. If a file name is not specified, the file name specified in the data rule is used. Format of each line:`startperiod=`*`START PERIOD`*`;endperiod=`*`END PERIOD`*`;importmode=IMPORT_MODE;exportmode=`*`EXPORT_MODE`*`;filename=`*`FILE NAME`*

A sample input file:

```
startperiod=Dec-15;endperiod=Dec-15;importmode=REPLACE;exportmode=STORE_DATA;f
ilename=comma_delim_file1.csv
startperiod=Dec-16;endperiod=Dec-16;importmode=REPLACE;exportmode=STORE_DATA;f
ilename=comma_delim_file2.csv
startperiod=Dec-17;endperiod=Dec-17;importmode=REPLACE;exportmode=STORE_DATA;f
ilename=comma_delim_file3.csv
startperiod=Dec-18;endperiod=Dec-18;importmode=REPLACE;exportmode=STORE_DATA;f
ilename=comma_delim_file4.csv
startperiod=Dec-19;endperiod=Dec-19;importmode=REPLACE;exportmode=STORE_DATA;f
ilename=comma_delim_file5.csv
```

Ad Hoc Grid Input File

The simulateConcurrentUsage command supports both native mode and standard mode ad hoc grids. In a native mode grids, the POVs are displayed as a part of the Oracle Smart View for Office plug-in tool bar. In standard mode, the POVs are a part of the spreadsheet itself and occupy the first row of the spreadsheet.

The ad hoc grid input file, referenced in `requirement.csv` to support the opening of Ad Hoc grids, should specify the grid to open. Each line of the file should be in the following format.

```
filename=xlsx filename#sheet name; pov=[DIM 1:MEMBER 1],[DIM 2:MEMBER 2]..;
rows=[ROW HEADER 1, ROW HEADER 2,..]; cols=[COL HEADER 1, COL HEADER 2,..]
```

A sample input file:

```
fileName=dropdown.xlsx#sheet4;pov=[HSP_View:BaseData],[Scenario:Forecast],
[Product:No Product],[Entity:Sales Mid-Atlantic];rows = [ Account ]; cols=
[Year, Period, Version]
```

> **Note:**
>
> The file identified by `fileName` in the input CSV file must contain the ad hoc grid definition in the specified sheet. For example, the preceding sample input file specifies `sheet4` of `dropdown.xlsx` as the source for the ad hoc grid definition. This Excel file must, along with `requirement.csv` and the input CSV file, be available in the *INPUT_FILE.zip* that is used to run the simulateConcurrentUsage command.

Execute Report Input File

This file, referenced in `requirement.csv` to support the opening of management reports, should specify the report to open. Each line of the file should be in the following format.

```
format=[REPORT_FORMAT];globalPov=[DIM 1:MEMBER 1],[DIM 2:MEMBER
2],..;prompts=[PROMPT 1:VALUE 1],[PROMPT 2:VALUE 2],..
```

`globalPov` and `prompts` are optional.

> **Note:**
>
> - Supported report formats are `pdf` and `embedded`.
> - If the name of the globalPov dimension or its members contain a colon (:) or semicolon (;), use the escape character \\ preceding it. For example, the dimension name `Version:View` should be specified as `Version\\:View`

A sample input file to generate a pdf from a report with the global POV `[Version View:Working],[Sales Entity:International Sales]` and prompts `[Actual;Budget], [Year:2018]`:

```
format=pdf;globalPov=[Version View:Working],[Sales Entity:International
Sales];prompts=[Actual:Budget],[Year:2018]
```

Execute Book Input File

This file, referenced in `requirement.csv` to support the opening of books in Reports, should specify the book to open. Each line of the file should be in the following format.

`format=BOOK_FORMAT` or

```
format=BOOK_FORMAT;globalPov=[DIM 1:MEMBER 1],[DIM 2:MEMBER 2],..
```

A sample input file to generate a pdf from a book with the global POV *[Version View:Working], [Sales Entity:International Sales]*:

```
format=pdf;globalPov=[Version View:Working],[Sales Entity:International Sales]
```

> **Note:**
>
> - Supported book formats are PDF and XLSX.
> - If the name of the globalPov dimension or its members contain a colon (:) or semicolon (;), use the escape character \\ preceding it. For example, the dimension name `Version:View` should be specified as `Version\\:View`

Creating the UserVarMemberMapping.csv File

This file is required if the forms you specified in the input file of the Open Form or Save Form use cases require user variables to be set. This file is not needed for other use cases.

The first line of this file is the header `#Dimension,User Variable,Member`

Subsequent entries contain the mapping of dimension, user variable and dimension member.

A sample `UserVarMemberMapping.csv` file:

```
#Dimension,User Variable,Member
Account,Account View,Revenue Driver Assumptions
Entity,Entity,No Entity
Entity,Entity View,Total Entity
HSP_View,HSP_View,BaseData
Market Size,Market View,Large Market
Period,Period,Jan
```

Creating the options.xml File

Oracle Smart View for Office options such as row suppression, missing block suppression, and page member indent can be used while simulating the Open Form, Save Form, and Ad Hoc Grid use cases. Each of these options exert different levels of load on the environment.

If you want to simulate the load using Smart View options, you can include the `options.xml` file in the input ZIP file.

You create `options.xml` by exporting Smart View options. To generate `options.xml`, from the **SmartView** tab, click **Options**, then **OK**, and then **Export Options**.

The use of `options.xml` is not mandatory. If this file is not present in the input ZIP file, the default options are used for the simulation.

Creating the users.csv File

Mode 4 of the simulateConcurrentUsage command allows you to run the command using existing users identified in the `users.csv` file included in the input ZIP file. This mode does not create users for simulation.

The format of the `users.csv` file is as follows:

```
loginname,password
jdoe,jdoe_pwd
john.doe@example.com,john_doe_pwd
```

The value of`loginname` in `users.csv` must match a login name defined in the identity domain. Also, the identified user must have the predefined and application roles necessary to run the operation.

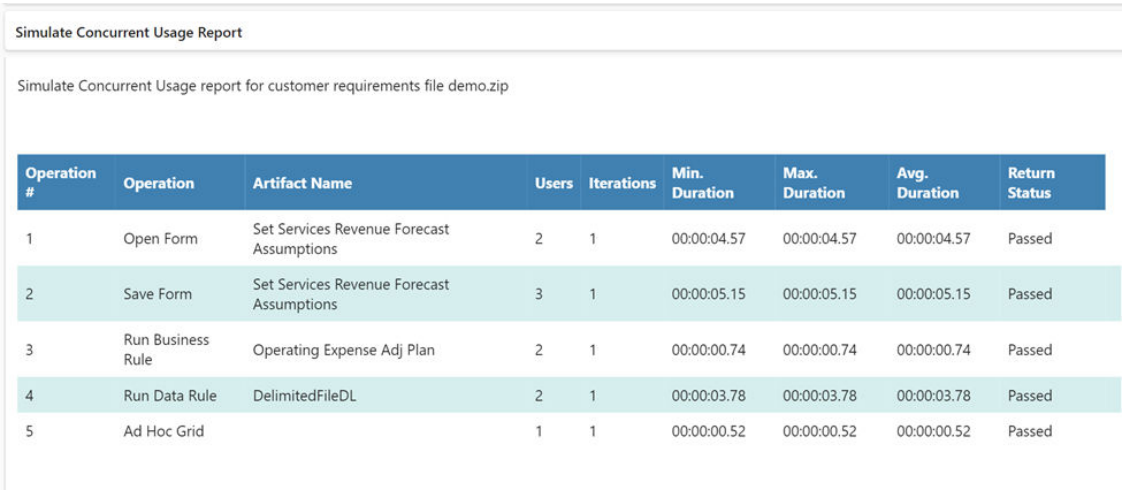
Creating and Uploading the Input ZIP File to the Environment

Using a tool such as 7 Zip, create one ZIP file containing `requirement.csv`, corresponding use case input files, and optionally, the `UserVarMemberMapping.csv`, `users.csv`, and `options.xml`, if required.

Use the uploadFile command to upload the resulting ZIP file into the inbox of the environment (example command syntax `epmautomate uploadFile "C:/uploads/INPUT_FILE.zip" inbox`) where you want to run the simulation.

Sample Simulate Concurrent Usage Report

The Simulate Concurrent Usage is, by default, sent to the user who executes the simulateConcurrentUsage command. If you specify email recipients, the report is emailed only to those email recipients.



Simulate Concurrent Usage Report

Simulate Concurrent Usage report for customer requirements file demo.zip

Operation #	Operation	Artifact Name	Users	Iterations	Min. Duration	Max. Duration	Avg. Duration	Return Status
1	Open Form	Set Services Revenue Forecast Assumptions	2	1	00:00:04.57	00:00:04.57	00:00:04.57	Passed
2	Save Form	Set Services Revenue Forecast Assumptions	3	1	00:00:05.15	00:00:05.15	00:00:05.15	Passed
3	Run Business Rule	Operating Expense Adj Plan	2	1	00:00:00.74	00:00:00.74	00:00:00.74	Passed
4	Run Data Rule	DelimitedFileDL	2	1	00:00:03.78	00:00:03.78	00:00:03.78	Passed
5	Ad Hoc Grid		1	1	00:00:00.52	00:00:00.52	00:00:00.52	Passed

This report identifies the following:

Column	Description
Operation #	The sequence number of the use case in `requirement.csv`
Operation	The type of the operation as specified in `requirement.csv`
Artifact Name	The artifact name as specified in `requirement.csv`
Users	The number of users as specified in `requirement.csv`

Column	Description
iterations	The number of times the use case was executed as specified by the `iterations` parameter
Min. Duration	The minimum time taken to execute this use case by one user
Max. Duration	The maximum time taken to execute this use case by one user
Avg. Duration	The average time taken to execute this use case by one user
Return Status	Status of the use case. `Failed` is shown if the use case execution was not successful

B

Preparing to Run the Replay Command

The replay command is used to performance test an environment under load to verify that user experience is acceptable when the service is under specified load. You need to complete a few steps before load testing environments.

This appendix describes the steps Service Administrators must complete before running the replay EPM Automate command.

About the Replay Command

The replay command replays the Oracle Smart View for Office, Oracle Enterprise Performance Management Cloud REST API, or EPM Automate load on an environment to enable performance testing under heavy load to verify that user experience is acceptable when the environment is under a specified load.

For example, you can test the user experience on a test environment under heavy load to ensure that it will perform well after you migrate the application from the test environment to the production environment.

Prerequisites

When you execute the command using a replay file, EPM Automate runs each row in the replay file in parallel to exert load on the service so that you can perform tests to verify that user experience is acceptable when the service is under load.

- Identify forms that require major processing on the environment. Forms that deal with large amounts of data, or forms that include complex calculations are good candidates. For example, forms that are used to submit forecast, processes involved in creating ad-hoc and static reports may exert heavy loads on the service. Similarly, activities such as running business rules, running reports, executing resource intensive REST APIs, and EPM Automate commands (for example, runBusiness rule, runDataRule, exportData, exportMetadata, restructureCube) my cause the environment to come under heavy load and can be candidates for load testing.
- Install Fiddler if necessary. EPM Automate requires an HTTP Archive format (HAR) 1.1 file containing records of Oracle Smart View for Office, Oracle Enterprise Performance Management Cloud REST API, or EPM Automate interaction with the EPM Cloud environment. Generally, you use Fiddler to generate the HAR file that captures the log of your interaction with EPM Cloud.

- Run the major activities that you identified previously. You use Smart View to run activities such as opening and saving forms, running business rules, and creating reports, and Fiddler to capture activity details and to export them to HAR files. Similarly, run REST APIs and EPM Automate commands and have Fiddler capture the details. See Creating HAR Files for details.
- Create a replay CSV file that lists the credentials (user names and passwords) and the name of the HAR files to run. Each row in the file may contain the user name and password of a unique user to simulate multiple simultaneous user sessions. See Creating Replay Files for details
  The user whose credentials are specified in a row to run a HAR file need not be the user who ran the session that was used to create the HAR file. However, this user should have the rights to run these activities on the environment.

See A Sample Replay Session for detailed steps to run the replay command.

Creating HAR Files

The HAR file captures traces of Oracle Smart View for Office, REST API, or EPM Automate interaction with Oracle Enterprise Performance Management Cloud.

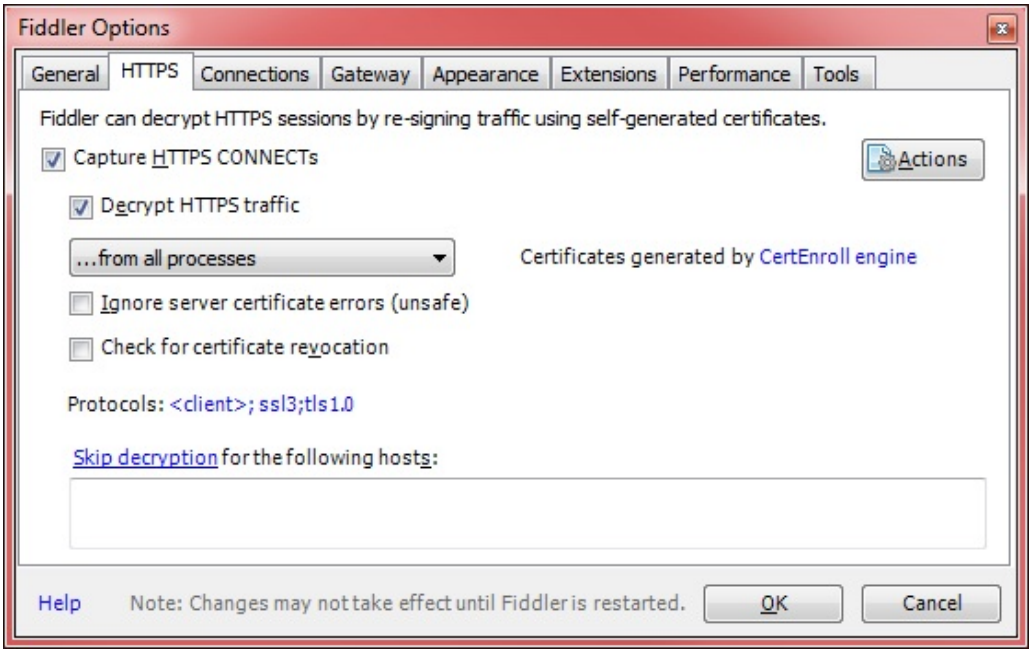
Because Fiddler captures information on all HTTP(S) traffic, while creating the HAR files, refrain from activities that may add unnecessary trace to Fiddler.

To create a HAR file:

1. Start Fiddler.
2. Ensure that Fiddler is configured to Decrypt HTTPS traffic from all processes.
   a. Select **Tools**, then **Options**, and then **HTTPS**.
   b. Select **Decrypt HTTPS traffic**, if it is not selected.

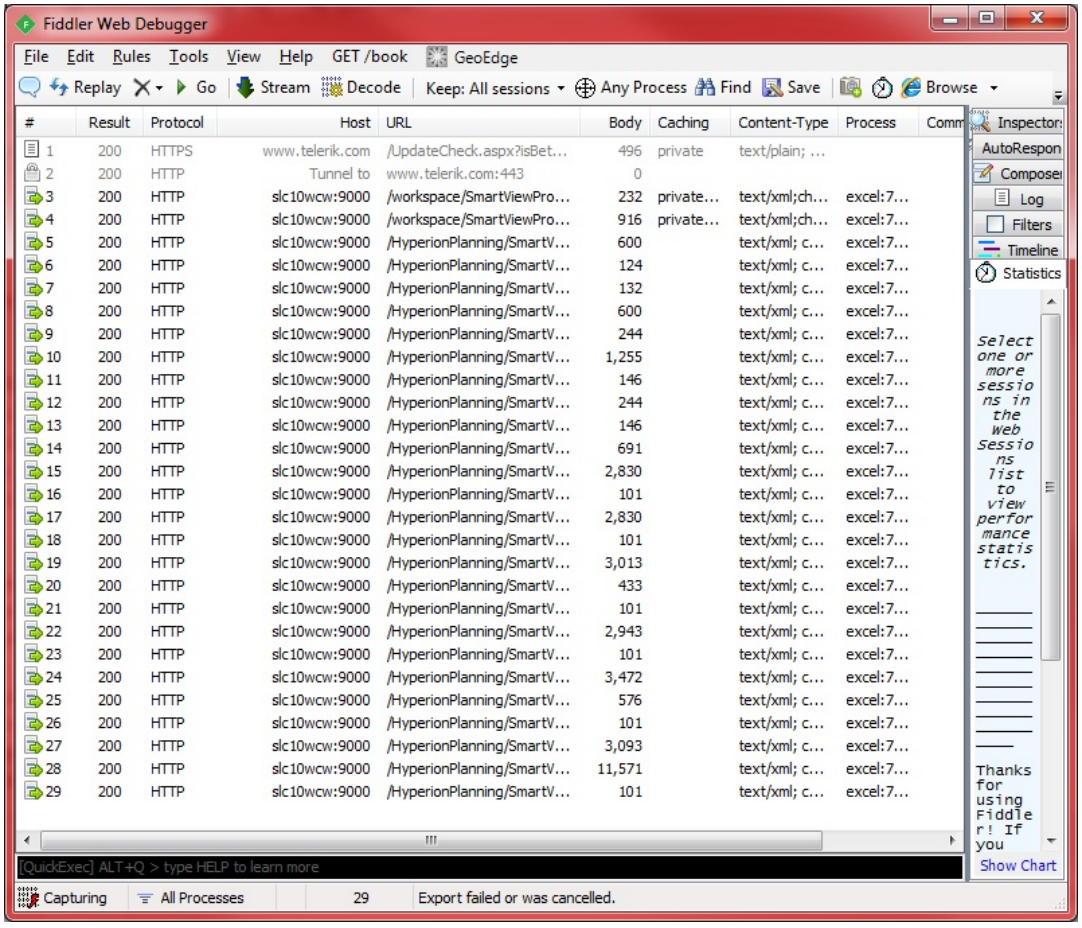
      Fiddler displays information about the root certificate it uses to intercept HTTPS traffic. It is generally safe to trust this certificate.

   c. Click **Yes** if you want to add the root certificate to the trusted CA list; else choose **No**.
   d. **Optional:** If you selected **No** in the preceding step, you may select **Ignore server certificate errors** to suppress Fiddler security warnings related to decrypting HTTPS traffic.
   e. Click **OK**.



3. Start Smart View and access the environment for which you want to capture trace.
4. Using Smart View, REST API, or EPM Automate, execute the activities that exert heavy processing load on the environment. For example, open forms in Smart View so that Fiddler can record your activity.

   Fiddler records the processes that you initiated.

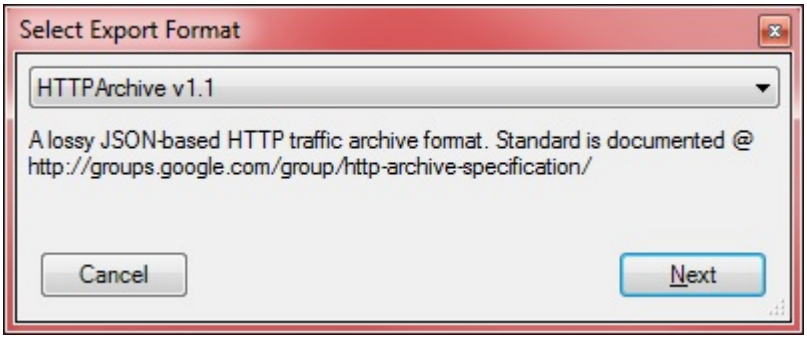


5. In Fiddler, complete these steps:

   a. Select `File`, then `Export Sessions`, and then either `All Sessions` or `Selected Sessions`. If you were connected to other web sites while running Fiddler, select `Selected Sessions` to choose the sessions relevant to the environment.

   b. In `Select Export Format`, select `HTTPArchive v1.1` as the export format.

   c. Click `Next`.



   d. In `Export As HTTPArcive v1.1`, select the directory where you want to store the file and specify a file name.

   e. Click `Save`.

Creating Replay Files

A replay file is a CSV file that lists the credentials (user name and password) and the name of the HAR files that are to be run to load the system using the replay EPM Automate command.

Ensure that the user name and password that you specify has the rights to run the activities included in the HAR file.

On executing the replay command, EPM Automate runs each row in the replay file in parallel to exert load on the service. For example, if your replay file contains 10 rows, EPM Automate replays 10 sessions so that you can perform tests to verify that user experience is acceptable when the service is under specified load. Each activity included in the HAR file is run serially.

See replay for information on running the replay command.

To create a replay file:

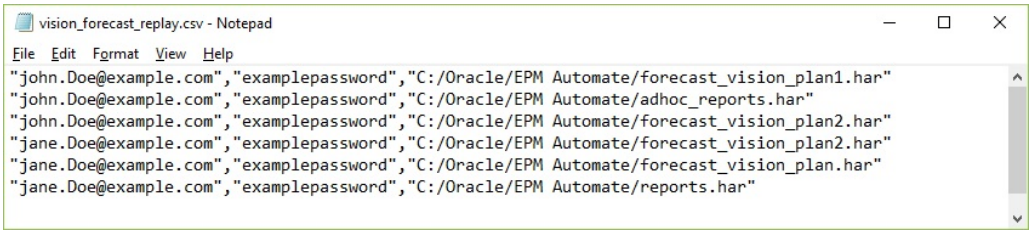
1. Open Microsoft Office Excel and start a new worksheet.
2. Enter a user name, password, and the location of a HAR file in Columns A, B, and C respectively of row 1.

   Repeat this step to create additional rows.

   > **Note:**
   >
   > You must specify the absolute path to the location of the HAR file. Use slash (/) as directory separator in file paths; do not use back slashes (\).

3. Save the file
4. In `Save As`, complete these steps:
   a. Select the directory where you want to store the replay file.
   b. In `File Name`, specify a name, and in `Save as type`, select `CSV (Comma delimited) (*.csv)`.
   c. Click `Save`.

      A sample replay file may be as follows:



Generating Trace Files

While running the replay command, you can generate trace files to share with Oracle Support to troubleshoot issues. Oracle Support uses trace files to understand how the environment handled an Oracle Smart View for Office activity.

You use the optional `trace=true` parameter with the replay command to generate trace files in XML format. If you use this parameter, for each activity in the HAR file, EPM Automate creates a trace file that contains Smart View response to the activity.

Trace files are named `trace-N.xml`; for example, `trace-1.xml` where `N` is a counter that starts at 1. If multiple identically named HAR files are specified in the replay file, EPM Automate consolidates the trace files in one folder.

Trace files related to a HAR file are stored in a folder within the directory from which you run EPM Automate. EPM Automate creates one folder for each HAR file listed in the replay file. EPM Automate uses a combination of current server system time and HAR file name in `YYYY_MM_DD_HH_MM_SS_HAR_FILE_NAME` format to name the folders. For example, if HAR file name is `forecast1.har`, the folder name may be `2016_06_08_10_21_42_forecast1`.

A Sample Replay Session

Describes how to run the replay command using multiple HAR files.

This section assumes the following:

- You created the following HAR files. Each HAR file may contain the same set of activities. See Creating HAR Files for detailed information.
  - `C:\Oracle\EPM Automate\forecast_vision_plan1.har`
  - `C:\Oracle\EPM Automate\forecast_vision_plan2.har`
  - `C:\Oracle\EPM Automate\forecast_plan2.har`
- You created a replay file `C:/Oracle/EPM Automate/vision_forecast_replay.csv` with the following content (see Creating Replay Files for details):

> **Note:**
>
> Use slash (/) as directory separator in file paths in the replay file; do not use back slashes (\).

```
john.doe@example.com,examplePwd,C:/Oracle/EPM Automate/
forecast_vision_plan1.har
john.doe@example.com,examplePwd,C:/Oracle/EPM Automate/
forecast_vision_plan2.har
john.doe@example.com,examplePwd,C:/Oracle/EPM Automate/forecast_plan2.har
```

To run the replay command:

1. In a Command Prompt window, navigate to the directory; for example, `C:\Oracle\EPM Automate\bin`, where EPM Automate is installed.

2. Sign in to an environment as a Service Administrator and then execute the replay command:

   ```
   epmautomate login john.doe@example.com examplePassword https://test-cloud-pln.pbcs.us1.oraclecloud.com myIdentityDomain
   ```

   ```
   epmautomate replay "c:/Oracle/EPM Automate/vision_forecast_replay.csv" duration=12 lagTime=5.5 trace=true
   ```

   EPM Automate displays replay information in the console and ends processing after the specified duration (12 minutes in the preceding example). It also creates trace folders and files because the preceding command includes the `trace=true` parameter.
   Because the command was executed from `C:\Oracle\EPM Automate\bin`, EPM Automate stored the trace files in the following folders. Note that these folders are named based on the HAR file names.

   - `C:\Oracle\EPM Automate\bin\2017_01_08-12_52_37-forecast_plan2-jdoe@example.com`
   - `C:\Oracle\EPM Automate\bin\2017_01_08-12_52_37-forecast_vision_plan1-jdoe@example.com`
   - `C:\Oracle\EPM Automate\bin\2017_01_08-12_52_37-forecast_vision_plan2-jdoe@example.com`

3. Sign out of the environment:

   ```
   epmautomate logout
   ```

C

Handling Special Characters

Oracle Enterprise Performance Management Cloud passwords, proxy passwords, and command parameter values may contain special characters. Special handling is required for EPM Automate to handle such characters.

The examples in this section use a sample password to illustrate the use of special characters.

Oracle recommends that you enclose parameter and value pairs in double quotation marks.

Windows

These special characters must be escaped using double quotation marks (") around the special character or around the parameter value containing the special character.

> **Note:**
>
> EPM Automate cannot be run from a folder that contains `&` in its name; for example, `C:\Oracle\A&B`.

Table C-1 Special Character Handling: Windows

Character	Description	Escaped Example
)	Close parenthesis	• `Example")"pwd1` or • `"Example)pwd1"`
<	Less than	• `Example"<"pwd1` or • `"Example<pwd1"`
>	Greater than	• `Example">"pwd1` or • `"Example>pwd1"`
&	Ampersand	• `Example"&"pwd1` or • `"Example&pwd1"`
|	Pipe	• `Example"|"pwd1` or • `"Example|pwd1"`
"	Quotation mark	• `Example""""pwd1` or • `"Example"pwd1"`

Using Exclamation Mark in Plain Text Passwords in Windows Batch Files

Use of exclamation mark (!) in plain text passwords in Windows batch files used with EPM Automate should be handled as follows:

1. Use two caret symbols (^^) before the exclamation mark as the escape character. For example, if the password is `Welc0me!`, encode it as `Welc0me^^!`
2. Update the bach file to set DisableDelayedExpansion at the beginning of the file by including the following declaration:

```
setlocal DisableDelayedExpansion
```

3. Remove `setlocal EnableExtensions EnableDelayedExpansion` declaration, if present, in the script.

UNIX/Linux

On UNIX and Linux operating systems, special characters must be escaped using a backslash (\).

> **Note:**
>
> - To escape `!` (exclamation mark), use a single quotation mark around the password or use the back slash (\) as the escape character.
> - To escape `\`, `$`, `'`, and `"`, use a double quotation mark around the password or use the back slash (\) as the escape character.

Table C-2 Special Character Handling: UNIX/Linux

Character	Description	Escaped Example
(	Open parenthesis	`Example\(pwd1`
)	Close parenthesis	`Example\)pwd1`
<	Less than	`Example\<pwd1`
>	Greater than	`Example\>pwd1`
`	Apostrophe	`Example\'pwd1`
!	exclamation mark	• `'Example!pwd1'` or • `Example\!pwd1`
#	Hash	`Example\#pwd1`
&	Ampersand	`Example\&pwd1`
|	Pipe	`Example|pwd1`
;	Semicolon	`Example\;pwd1`
.	Period	`Example\.pwd1`
"	Quotation mark	• `Example\"pwd1` or • `"Example\"pwd1"`
'	Single quotation mark	• `Example\'pwd1` or • `"Example\'pwd1"`
$	Dollar sign	• `Example\$pwd1` or • `"Example\$pwd1"`
\	Back slash	• `Example\\pwd1` or • `"Example\\pwd1"`

Using Exclamation Mark in Plain Text Passwords in UNIX or Linux Scripts

In UNIX/Linux scripts, if an EPM Automate password stored in a shell variable contains special characters, use three back slashes as the escape sequence and then enclose the string in

double quotation marks. For example, the password `lzi[ACO(e*7Qd)jE` included in the shell variable `password` should be scripted as follows:

```
password="lzi[ACO\\\(e*7Qd\\\)jE"
```

D

Commands Specific to Each EPM Cloud Service

Account Reconciliation Commands

EPM Automate Commands for Account Reconciliation		
addUsers	groupAssignmentAuditReport	renameSnapshot
addUsersToGroup	help	replay
addUsersToTeam	importARApplicationProperties	resetService
addUserToGroups	importBackgroundImage	restoreBackup
archiveTmTransactions	importLogoImage	roleAssignmentAuditReport
assignRole	importBalances	roleAssignmentReport
cloneEnvironment	importDataManagement	runAutomatch
copyFileFromInstance	importMapping	runBatch
copyFromObjectStorage	importPreMappedBalances	runComplianceReport
copySnapshotFromInstance	importPreMappedTransactions	runDailyMaintenance
copyToObjectStorage	importProfiles	runDataRule
createGroups	importRates	runDMReport
createReconciliations	importRCAttributeValues	runIntegration
deleteFile	importReconciliationAttributes	runMatchingReport
deleteGroups	importSnapshot	sendMail
downloadFile	importTMAttributeValues	setApplicationAdminMode
encrypt	importTmPremappedTransactions	setDailyMaintenanceStartTime
exportAccessControl	invalidLoginReport	setDemoDates
exportARApplicationProperties	listBackups	setEncryptionKey
exportBackgroundImage	listFiles	setIdleSessionTimeout
exportDataManagement	login	setIPAllowlist
exportLogoImage	logout	setRestrictedDataAccess
exportMapping	provisionReport	setManualDataAccess
exportSnapshot	purgeArchivedTmTransactions	setPeriodStatus
feedback	purgeTmTransactions	setVirusScanOnFileUploads
getApplicationAdminMode	recreate	skipUpdate
getDailyMaintenanceStartTime	refreshCube	unassignRole
getIdleSessionTimeout	removeUserFromGroups	updateUsers
getIPAllowlist	removeUsers	upgrade
getRestrictedDataAccess	removeUsersFromGroup	uploadFile
getVirusScanOnFileUploads	removeUsersFromTeam	userAuditReport
		userGroupReport

Financial Consolidation and Close Commands

EPM Automate Commands for Financial Consolidation and Close

addUsers
addUsersToGroup
addUsersToTeam
addUserToGroups
applicationAdminMode
assignRole
clearDataByProfile
cloneEnvironment
copyDataByProfile
copyFileFromInstance
copyFromObjectStorage
copyOwnershipDataToNextYear
copySnapshotFromInstance
copyToObjectStorage
createGroups
deleteFile
deleteGroups
deployEJTemplates
deployFormTemplates
deployTaskManagerTemplate
downloadFile
essbaseBlockAnalysisReport
executeReportBurstingDefinition
exportDataManagement
exportEssbaseData
encrypt
exportAppAudit
exportAppSecurity
exportConsolidationJournals
exportData
exportEJJournals
exportJobConsole
exportLibraryDocument
exportMapping
exportMetadata
exportOwnershipData
exportSnapshot
exportTaskManagerAccessControl
exportValidIntersections
exportSnapshot
exportTaskManagerAccessControl
exportValidIntersections
feedback
getApplicationAdminMode
getDailyMaintenanceStartTime
getEssbaseQryGovExecTime
getIdleSessionTimeout
getIPAllowlist
getRestrictedDataAccess
getSubstVar
getVirusScanOnFileUploads
groupAssignmentAuditReport
help
importAppSecurity
importConsolidationJournals
importData
importDataManagement
importJobConsole
importMapping
importMetadata
importOwnershipData
importSnapshot
importSupplementalCollectionData
importSupplementalData
importValidIntersections
invalidLoginReport
listBackups
listFiles
login
logout
maskData
provisionReport
recomputeOwnershipData
recreate
refreshCube
removeUserFromGroups
removeUsers
removeUsersFromGroup
removeUsersFromTeam
renameSnapshot
replay
resetService
restoreBackup
restructureCube
roleAssignmentAuditReport
roleAssignmentReport
runBatch
runBusinessRule
runDailyMaintenance
runDataRule
runDMReport
runIntegration
runRuleSet
runIntercompanyMatchingReport
runSupplementalDataReport
runTaskManagerReport
sendMail
setApplicationAdminMode
setDailyMaintenanceStartTime
setDemoDates
setEJJournalStatus
setEncryptionKey
setEssbaseQryGovExecTime
setIdleSessionTimeout
setIPAllowlist
setRestrictedDataAccess
setVirusScanOnFileUploads
setManualDataAccess
setSubstVars
simulateConcurrentUsage
skipUpdate
snapshotCompareReport
unassignRole
updateUsers
upgrade
uploadFile
userAuditReport
userGroupReport
validateConsolidationMetadata

Narrative Reporting Commands

EPM Automate Commands for Narrative Reporting		
addUsers	getIdleSessionTimeout	restoreBackup
addUsersToGroup	getIPAllowlist	roleAssignmentAuditReport
addUserToGroups	getRestrictedDataAccess	roleAssignmentReport
assignRole	getVirusScanOnFileUploads	runDailyMaintenance
cloneEnvironment	groupAssignmentAuditReport	sendMail
copyFileFromInstance	help	setDailyMaintenanceStartTime
copyFromObjectStorage	importLibraryArtifact	setEncryptionKey
copyToObjectStorage	invalidLoginReport	setIdleSessionTimeout
createGroups	listBackups	setIPAllowlist
createNRSnapshot	listFiles	setManualDataAccess
deleteFile	login	setRestrictedDataAccess
deleteGroups	logout	setVirusScanOnFileUploads
downloadFile	provisionReport	skipUpdate
encrypt	recreate	unassignRole
executeBurstDefinition	removeUserFromGroups	updateUsers
exportLibraryArtifact	removeUsers	upgrade
feedback	removeUsersFromGroup	uploadFile
getDailyMaintenanceStartTime	replay	userAuditReport
	resetService	userGroupReport

Oracle Enterprise Data Management Cloud Commands

EPM Automate Commands for Oracle Enterprise Data Management Cloud		
addUsers	getIdleSessionTimeout	replay
addUsersToGroup	getIPAllowlist	resetService
addUserToGroups	getRestrictedDataAccess	restoreBackup
assignRole	getVirusScanOnFileUploads	roleAssignmentAuditReport
cloneEnvironment	groupAssignmentAuditReport	roleAssignmentReport
copyFileFromInstance	help	runDailyMaintenance
copyFromObjectStorage	importDimension	sendMail
copySnapshotFromInstance	importSnapshot	setDailyMaintenanceStartTime
copyToObjectStorage	invalidLoginReport	setEncryptionKey
createGroups	listBackups	setIdleSessionTimeout
deleteFile	listFiles	setIPAllowlist
deleteGroups	loadViewpoint	setRestrictedDataAccess
downloadFile	login	setManualDataAccess
encrypt	logout	setVirusScanOnFileUploads
exportDimension	provisionReport	skipUpdate
exportDimensionMapping	recreate	unassignRole
exportSnapshot	removeUserFromGroups	updateUsers
extractDimension	removeUsers	upgrade
extractPackage	removeUsersFromGroup	uploadFile
feedback	renameSnapshot	userAuditReport
getDailyMaintenanceStartTime		userGroupReport

Planning, Planning Modules, FreeForm, Strategic Workforce Planning, and Sales Planning Commands

EPM Automate Commands for Planning, Planning Modules, FreeForm, Strategic Workforce Planning, and Sales Planning

addUsers
addUsersToGroup
addUserToGroups
applicationAdminMode
assignRole
autoPredict * See footnote
clearCube
cloneEnvironment
copyFileFromInstance
copyFromObjectStorage
copySnapshotFromInstance
copyToObjectStorage
createGroups
deleteFile
deleteGroups
dismissIPMInsights**
downloadFile
enableQueryTracking
encrypt
essbaseBlockAnalysisReport
executeAggregationProcess
executeReportBurstingDefinition
exportAppAudit
exportAppSecurity
exportCellLevelSecurity
exportData
exportDataManagement
exportEssbaseData
exportJobConsole
exportLibraryDocument
exportMapping
exportMetadata
exportSnapshot
exportValidIntersections
feedback
getApplicationAdminMode
getDailyMaintenanceStartTime
getEssbaseQryGovExecTime
getIdleSessionTimeout
getIPAllowlist
getRestrictedDataAccess
getSubstVar
getVirusScanOnFileUploads
groupAssignmentAuditReport
help
importAppAudit
importAppSecurity
importCellLevelSecurity
importData
importDataManagement
importJobConsole
importMapping
importMetadata
importSnapshot
importValidIntersections
invalidLoginReport
listBackups
listFiles
login
logout
maskData
mergeDataSlices
provisionReport
recreate
refreshCube
removeUserFromGroups
removeUsers
removeUsersFromGroup
renameSnapshot
replay
resetService
restoreBackup
restructureCube
roleAssignmentAuditReport
roleAssignmentReport
runBatch
runBusinessRule
runDailyMaintenance
runDataRule
runDMReport
runIntegration
runPlanTypeMap
runRuleSet
sendMail
setApplicationAdminMode
setDailyMaintenanceStartTime
setEncryptionKey
setEssbaseQryGovExecTime
setIdleSessionTimeout
setIPAllowlist
setManualDataAccess
setRestrictedDataAccess
setSubstVars
setVirusScanOnFileUploads
simulateConcurrentUsage
skipUpdate
snapshotCompareReport
sortMember
unassignRole
updateUsers
upgrade
uploadFile
userAuditReport
userGroupReport

* This command is not supported for FreeForm, Strategic Workforce Planning and Sales Planning.
** This command is not supported for FreeForm.

Profitability and Cost Management Commands

EPM Automate Commands for Profitability and Cost Management		
addUsers	getEssbaseQryGovExecTime	renameSnapshot
addUsersToGroup	getIdleSessionTimeout	replay
addUserToGroups	getIPAllowlist	resetService
applyDataGrants	getRestrictedDataAccess	restoreBackup
assignRole	getVirusScanOnFileUploads	roleAssignmentAuditReport
clearPOV	groupAssignmentAuditReport	roleAssignmentReport
cloneEnvironment	help	runBatch
copyFileFromInstance	importDataManagement	runCalc
copyFromObjectStorage	importMapping	runDailyMaintenance
copyPOV	importSnapshot	runDataRule
copySnapshotFromInstance	importTemplate	runDMReport
copyToObjectStorage	invalidLoginReport	runIntegration
createGroups	listBackups	sendMail
deleteFile	listFiles	setDailyMaintenanceStartTime
deleteGroups	loadData	setEncryptionKey
deletePOV	loadDimData	setEssbaseQryGovExecTime
deployCube	login	setIdleSessionTimeout
downloadFile	logout	setIPAllowlist
enableApp	mergeSlices	setRestrictedDataAccess
encrypt	optimizeASOCube	setManualDataAccess
exportDataManagement	programDocumentationReport	skipUpdate
exportMapping	provisionReport	setVirusScanOnFileUploads
exportQueryResults	recreate	unassignRole
exportSnapshot	removeUserFromGroups	updateUsers
exportTemplate	removeUsers	upgrade
feedback	removeUsersFromGroup	uploadFile
getDailyMaintenanceStartTime		userAuditReport
		userGroupReport

Enterprise Profitability and Cost Management Commands

EPM Automate Commands for Enterprise Profitability and Cost Management		
addUsers	exportValidIntersections	removeUsers
addUsersToGroup	feedback	removeUsersFromGroup
addUserToGroups	getApplicationAdminMode	renameSnapshot
applicationAdminMode	getDailyMaintenanceStartTime	replay
assignRole	getEssbaseQryGovExecTime	resetService
calculateModel	getIdleSessionTimeout	restoreBackup
clearCube	getIPAllowlist	roleAssignmentAuditReport
copyDataByPointOfView	getRestrictedDataAccess	roleAssignmentReport
cloneEnvironment	getSubstVar	runBatch
copyFileFromInstance	getVirusScanOnFileUploads	runDailyMaintenance
copyFromObjectStorage	groupAssignmentAuditReport	runDataRule
clearDataByPointOfView	help	runDMReport
copySnapshotFromInstance	importAppAudit	runIntegration
copyToObjectStorage	importAppSecurity	sendMail
createGroups	importCellLevelSecurity	setApplicationAdminMode
deleteFile	importData	setDailyMaintenanceStartTime
deleteGroups	importDataManagement	setEncryptionKey
downloadFile	importJobConsole	setEssbaseQryGovExecTime
deletePointOfView	importMapping	setIdleSessionTimeout
enableQueryTracking	importMetadata	setIPAllowlist
encrypt	importSnapshot	setManualDataAccess
executeAggregationProcess	importValidIntersections	setRestrictedDataAccess
executeReportBurstingDefinition	invalidLoginReport	setSubstVars
exportAppAudit	listBackups	setVirusScanOnFileUploads
exportAppSecurity	listFiles	skipUpdate
exportCellLevelSecurity	login	snapshotCompareReport
exportData	logout	sortMember
exportDataManagement	maskData	unassignRole
exportEssbaseData	mergeDataSlices	updateUsers
exportJobConsole	provisionReport	upgrade
exportLibraryDocument	recreate	uploadFile
exportMapping	refreshCube	userAuditReport
exportMetadata	removeUserFromGroups	userGroupReport
exportMetadata		validateModel
exportSnapshot		

Tax Reporting Commands

EPM Automate Commands for Tax Reporting		
addUsers	getDailyMaintenanceStartTime	replay
addUsersToGroup	getEssbaseQryGovExecTime	resetService
addUsersToTeam	getIdleSessionTimeout	restoreBackup
addUserToGroups	getIPAllowlist	restructureCube
applicationAdminMode	getRestrictedDataAccess	roleAssignmentAuditReport
assignRole	getSubstVar	roleAssignmentReport
clearDataByProfile	getVirusScanOnFileUploads	runBatch
copyDataByProfile	groupAssignmentAuditReport	runBusinessRule
copyFileFromInstance	help	runDailyMaintenance
copyFromObjectStorage	importAppSecurity	runDataRule
copyOwnershipDataToNextYear	importCellLevelSecurity	runDMReport
copySnapshotFromInstance	importData	runIntegration
copyToObjectStorage	importDataManagement	runRuleSet
createGroups	importJobConsole	runSupplementalDataReport
deleteFile	importMapping	runTaskManagerReport
deleteGroups	importMetadata	sendMail
deployFormTemplates	importOwnershipData	setApplicationAdminMode
downloadFile	importSnapshot	setDailyMaintenanceStartTime
encrypt	importSupplementalCollectionData	setDemoDates
essbaseBlockAnalysisReport	importSupplementalData	setEncryptionKey
executeReportBurstingDefinition	importValidIntersections	setEssbaseQryGovExecTime
exportAppAudit	invalidLoginReport	setIdleSessionTimeout
exportCellLevelSecurity	listBackups	setIPAllowlist
exportData	listFiles	setManualDataAccess
exportDataManagement	login	setRestrictedDataAccess
exportEssbaseData	logout	setSubstVars
exportJobConsole	maskData	setVirusScanOnFileUploads
exportLibraryDocument	provisionReport	simulateConcurrentUsage
exportMapping	recomputeOwnershipData	skipUpdate
exportMetadata	recreate	snapshotCompareReport
exportOwnershipData	refreshCube	unassignRole
exportSnapshot	removeUserFromGroups	upgrade
exportTaskManagerAccessControl	removeUsers	updateUsers
exportValidIntersections	removeUsersFromGroup	uploadFile
feedback	removeUsersFromTeam	userAuditReport
getApplicationAdminMode	renameSnapshot	userGroupReport