Oracle® Fusion Cloud EPM

使用 EPM Automate

F28912-24

Oracle Fusion Cloud EPM 使用 EPM Automate

F28912-24

第一作者：EPM Information Development Team

目录

文档可访问性

文档反馈

1 关于 EPM Automate

安装 EPM Automate 1-1
- 容量和端口要求 1-2
- 支持的平台 1-2
- Java 运行时环境和 EPM Automate 1-3
- 使用 OpenJDK 1-3
- Windows 说明 1-4
- Linux/UNIX/macOS X 说明 1-4
- EPM Automate 命令的服务器端执行 1-5

更新 EPM Automate 1-5
卸载 EPM Automate 1-5
了解 EPM Automate 加密级别 1-6
将 OAuth 2.0 授权协议用于 OCI（第 2 代）环境 1-6

2 命令参考

关于运行 EPM Automate 命令 2-1
- 先决条件 2-1
- 默认文件位置 2-2
- 启用传输层安全协议 1.2 2-3
- 使用 EPM Automate 命令 2-4
- 为一个参数指定多个值 2-4
- 日常维护期间的行为 2-4

运行 EPM Automate 2-5
- Windows 2-5
- Linux 2-6
- 运行 EPM Automate 的多个实例 2-7

命令概览 2-9
EPM Automate 命令 2-14
addUsers 2-14
addUsersToGroup 2-15
addUsersToTeam 2-16
addUserToGroups 2-17
applicationAdminMode 2-18
applyDataGrants 2-18
archiveTmTransactions 2-19
assignRole 2-20
autoPredict 2-22
calculateModel 2-22
clearCube 2-24
clearDataByPointOfView 2-25
clearDataByProfile 2-26
clearPOV 2-26
cloneEnvironment 2-27
compactCube 2-31
copyDataByPointOfView 2-31
copyDataByProfile 2-32
copyFileFromInstance 2-32
copyFromObjectStorage 2-33
copyFromSFTP 2-35
copyOwnershipDataToNextYear 2-36
copyPOV 2-36
copySnapshotFromInstance 2-37
copyToObjectStorage 2-38
copyToSFTP 2-40
createGroups 2-40
createNRSnapshot 2-41
createReconciliations 2-41
deleteFile 2-42
deleteGroups 2-43
deletePointOfView 2-44
deletePOV 2-44
deployCube 2-45
deployEJTemplates 2-46
deployFormTemplates 2-46
deployTaskManagerTemplate 2-47
dismissIPMInsights 2-48
downloadFile 2-48
enableApp 2-49

enableQueryTracking 2-49
encrypt 2-50
essbaseBlockAnalysisReport 2-51
executeAggregationProcess 2-52
executeBurstDefinition 2-53
executeReportBurstingDefinition 2-53
exportAccessControl 2-54
exportAppAudit 2-54
exportAppSecurity 2-55
exportARApplicationProperties 2-56
exportBackgroundImage 2-57
exportCellLevelSecurity 2-57
exportConsolidationJournals 2-58
exportData 2-58
exportDataManagement 2-58
exportDimension 2-59
exportDimensionMapping 2-60
exportEJJournals 2-61
exportEssbaseData 2-62
exportJobConsole 2-62
exportLibraryArtifact 2-65
exportLibraryDocument 2-66
exportLogoImage 2-67
exportMapping 2-67
exportMetadata 2-68
exportOwnershipData 2-69
exportQueryResults 2-69
exportSnapshot 2-71
exportTemplate 2-72
exportTaskManagerAccessControl 2-72
exportValidIntersections 2-73
extractDimension 2-74
extractPackage 2-75
feedback 2-75
getApplicationAdminMode 2-76
getDailyMaintenanceStartTime 2-77
getEssbaseQryGovExecTime 2-78
getIdleSessionTimeout 2-78
getIPAllowlist 2-79
getRestrictedDataAccess 2-79
getSubstVar 2-80
getVirusScanOnFileUploads 2-81

groupAssignmentAuditReport 2-81
help 2-82
importAppAudit 2-82
importAppSecurity 2-83
importARApplicationProperties 2-84
importBackgroundImage 2-84
importBalances 2-85
importCellLevelSecurity 2-85
importConsolidationJournals 2-86
importData 2-87
importDataManagement 2-87
importDimension 2-88
importJobConsole 2-89
importLibraryArtifact 2-90
importLogoImage 2-91
importMapping 2-91
importMetadata 2-92
importOwnershipData 2-93
importPreMappedBalances 2-94
importPreMappedTransactions 2-94
importProfiles 2-95
importRates 2-96
importRCAttributeValues 2-96
importReconciliationAttributes 2-97
importSnapshot 2-98
importSupplementalCollectionData 2-100
importSupplementalData 2-101
importTemplate 2-102
importTMAttributeValues 2-103
importTmPremappedTransactions 2-104
importValidIntersections 2-105
invalidLoginReport 2-106
listBackups 2-107
listFiles 2-108
loadData 2-109
loadDimensionViewpoint 2-109
loadDimData 2-110
loadViewpoint 2-111
login 2-111
logout 2-115
maskData 2-115
mergeDataSlices 2-116

mergeSlices 2-116
optimizeASOCube 2-117
programDocumentationReport 2-118
provisionReport 2-118
purgeArchivedTmTransactions 2-120
purgeTmTransactions 2-120
recomputeOwnershipData 2-121
recreate 2-122
refreshCube 2-125
removeUserFromGroups 2-125
removeUsers 2-126
removeUsersFromGroup 2-127
removeUsersFromTeam 2-128
renameSnapshot 2-129
replay 2-129
resetService 2-131
restoreBackup 2-131
restructureCube 2-132
roleAssignmentAuditReport 2-133
roleAssignmentReport 2-134
runAutomatch 2-136
runBatch 2-136
runBusinessRule 2-137
runCalc 2-138
runComplianceReport 2-139
runDailyMaintenance 2-140
runDataRule 2-141
runDMReport 2-142
runIntegration 2-143
runIntercompanyMatchingReport 2-146
runMatchingReport 2-147
runPipeline 2-148
runPlanTypeMap 2-150
runRuleSet 2-150
runSupplementalDataReport 2-151
runTaskManagerReport 2-152
sendMail 2-153
setApplicationAdminMode 2-154
setDailyMaintenanceStartTime 2-155
setDemoDates 2-155
setEJJournalStatus 2-156
setEncryptionKey 2-157

setEssbaseQryGovExecTime 2-158
setIdleSessionTimeout 2-158
setIPAllowlist 2-159
setManualDataAccess 2-160
setPeriodStatus 2-161
setRestrictedDataAccess 2-162
setSubstVars 2-162
setVirusScanOnFileUploads 2-163
simulateConcurrentUsage 2-163
skipUpdate 2-167
snapshotCompareReport 2-168
sortMember 2-169
unassignRole 2-170
updateGuidedLearningSettings 2-171
updateUsers 2-172
upgrade 2-173
uploadFile 2-174
userAuditReport 2-175
userGroupReport 2-176
validateConsolidationMetadata 2-176
validateModel 2-177
退出代码 2-178

3 命令执行示例方案

关于复制示例脚本 3-1
所有服务的示例方案 3-1
将应用程序快照备份到计算机 3-2
向用户发送完成日常维护的通知 3-4
将快照复制到 Oracle Object Storage 或从中复制快照 3-12
创建用户并为其分配预定义角色 3-14
统计许可的用户数（分配了角色的用户） 3-17
创建分配了角色的用户的审核报表 3-19
创建角色分配和撤消审核报表 3-23
遮蔽访问日志和活动报表以确保符合隐私法 3-26
自动将活动报表下载到本地计算机 3-31
从环境中下载访问日志 3-35
自动克隆环境 3-38
在主环境完成日常维护后，每天从主环境克隆到备用环境 3-41
从环境中删除不必要的文件 3-50
从环境中查找并下载文件 3-52
重新创建一个旧云 EPM 环境用于审核 3-53

自动进行数据库访问审核与合规性调节 3-64
复制用户和预定义角色分配 3-74
将一个身份域的用户复制到另一个身份域 3-75
将预定义角色分配从一个环境复制到另一个环境 3-80
创建每季度一次的云 EPM 升级节奏 3-88
Windows 脚本和说明 3-89
UNIX/Linux 脚本和说明 3-92
Groovy 脚本 3-95
创建每季度一次的云 EPM 升级节奏，并实施为期六周的测试周期 3-99
Planning、Consolidation、Tax Reporting 和 Enterprise Profitability and Cost Management 的示例方案 3-112
自动从聚合存储多维数据集导出大量单元格 3-113
将元数据导入应用程序 3-122
导入数据、运行计算脚本并将数据从块存储数据库复制到聚合存储数据库 3-123
导出和下载元数据和数据 3-126
导出和下载应用程序数据 3-128
自动存档应用程序审核记录 3-130
Windows 脚本 3-131
Linux 脚本 3-132
将数据文件上传到环境并运行数据加载规则 3-133
自动执行每日数据集成 3-136
Account Reconciliation 示例方案 3-138
将预先设置格式的余额加载到期间内 3-138
上传和导入备份快照 3-140
存档旧的匹配事务和清除存档的事务 3-142
Profitability and Cost Management 示例方案 3-147
将元数据导入应用程序中 3-148
导入数据并运行程序规则 3-150
Oracle Enterprise Data Management Cloud 示例方案 3-152
将 Oracle Enterprise Data Management Cloud 维和映射与云 EPM 应用程序同步 3-152
将云 EPM 维与 Oracle Enterprise Data Management Cloud 应用程序同步 3-154
自动执行脚本 3-155
监视 EPM Automate 活动 3-156

4 在不安装 EPM Automate 的情况下运行命令

支持服务器端命令执行的环境 4-1
信息源 4-1
支持的命令 4-2
使用服务器端 Groovy 运行 EPM Automate 的方法 4-2
使用服务器端 Groovy 脚本克隆环境 4-3

使用服务器端 Groovy 脚本通过电子邮件发送活动报表 4-4

5 复制云 EPM 环境

设置每日复制 5-1
设置按需复制 5-2
配置辅助环境 5-2

A 准备运行 simulateConcurrentUsage 命令

创建 requirement.csv 文件 A-1
创建输入文件 A-3
打开表单输入文件 A-3
保存表单输入文件 A-4
运行业务规则输入文件 A-5
运行业务规则集输入文件 A-5
运行数据规则输入文件 A-5
即席网格输入文件 A-6
执行报表输入文件 A-6
执行工作簿输入文件 A-7
创建 UserVarMemberMapping.csv 文件 A-7
创建 options.xml 文件 A-7
创建 users.csv 文件 A-8
创建输入 ZIP 文件并将其上传到环境 A-8
"Simulate Concurrent Usage Report"（模拟并发使用报表）示例 A-8

B 准备运行 Replay 命令

关于 Replay 命令 B-1
先决条件 B-1
创建 HAR 文件 B-2
创建重放文件 B-5
生成跟踪文件 B-5
重放示例会话 B-6

C 处理特殊字符

D 特定于每个云 EPM 服务的命令

Account Reconciliation 命令 D-2
Financial Consolidation and Close 命令 D-3

Narrative Reporting 命令 D-4
Oracle Enterprise Data Management Cloud 命令 D-5
Planning、Planning 模块、自由形式、战略性人员规划和销售规划命令 D-6
Profitability and Cost Management 命令 D-7
Enterprise Profitability and Cost Management 命令 D-8
Tax Reporting 命令 D-9

文档可访问性

有关 Oracle 对可访问性的承诺，请访问 Oracle Accessibility Program 网站 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc。

获得 Oracle 支持

购买了支持服务的 Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

文档反馈

要提供有关此文档的反馈，请单击任意 Oracle 帮助中心主题中页面底部的“反馈”按钮。还可以向 epmdoc_ww@oracle.com 发送电子邮件。

1

关于 EPM Automate

使用 EPM Automate，用户可以在 Oracle Fusion Cloud Enterprise Performance Management 环境中远程执行任务。

云 EPM 服务管理员可以自动执行许多可重复的任务（包括以下任务）：

- 导入和导出元数据、数据、对象和应用程序快照、模板以及数据管理映射
- 将文件上传到环境，列出文件并将文件从服务删除
- 从服务下载快照、报表以及元数据和数据文件
- 对数据运行业务规则，并刷新应用程序
- 将数据从一个数据库复制到另一个数据库；通常，从块存储数据库复制到聚合存储数据库，或者从一个块存储数据库复制到另一个块存储数据库
- 运行数据管理批处理规则
- 生成数据管理报表、设置报表和用户审核报表
- 将预映射的余额数据、汇率、预映射的事务、余额数据和配置文件导入 Account Reconciliation
- 将配置文件复制到期间以启动调节流程
- 部署 Profitability and Cost Management 应用程序的计算多维数据集
- 清除、复制和删除 Enterprise Profitability and Cost Management 和 Profitability and Cost Management 应用程序中的视点
- 在环境中重放 Oracle Smart View for Office 或 REST API 负载以在重负载下进行性能测试
- 将补充数据从文件导入到 Financial Consolidation and Close

可以创建能够完成各种任务的脚本并通过计划程序自动执行这些脚本。例如，您可以创建一个脚本，从环境下载日常维护备份以创建对象和数据的本地备份。

教程：如何使用 EPM Automate 执行 Planning 命令

安装 EPM Automate

您可以安装 EPM Automate 以运行命令。此外，还可以使用 Groovy 脚本直接在 Oracle Fusion Cloud Enterprise Performance Management 中运行某些命令，而无需安装 EPM Automate。

适用于 Windows、Linux/UNIX 和 macOS X 的 EPM Automate 安装程序可以从您的云 EPM 环境中获取。

因为 Windows 10 和更高版本仅允许 Windows 管理员安装 EPM Automate，所以它只能由 Windows 管理员安装和升级。EPM Automate 可以由安装它的用户或其他 Windows 管理员升级。

在本节中：

- 容量和端口要求

- 支持的平台
- Java 运行时环境和 EPM Automate
- 使用 OpenJDK
- Windows 说明
- Linux/UNIX/macOS X 说明
- EPM Automate 命令的服务器端执行

容量和端口要求

由于 EPM Automate 是轻型客户端，它不需要占用较大的客户端空间。所有处理都发生在 Oracle Fusion Cloud Enterprise Performance Management 中。

您可以将 EPM Automate 安装在标准客户端计算机上、虚拟机上和可以通过安全 HTTP 连接访问外部主机的 Oracle Integration Cloud 计算机上。

EPM Automate 使用标准 TLS 端口（端口 443）连接到云 EPM。无需为 EPM Automate 打开其他传出端口。

此时，EPM Automate 不支持相互 TLS (mTLS) 身份验证。

支持的平台

EPM Automate 可安装在虚拟机上和可以通过安全 HTTP 连接访问外部主机的 Oracle Integration Cloud (OIC) 计算机上。

> **Note:**
>
> - EPM Automate 只能在操作系统供应商当前支持的 64 位操作系统上使用。
> - EPM Automate 不能与 SOCKS 代理一起使用，只能与 HTTP/HTTPS 代理一起使用。
> - EPM Automate 支持您使用基本、摘要、Kerberos、协商和 NTLM 身份验证机制连接到代理服务器。
> - EPM Automate 可以通过 API 网关连接到 Oracle Fusion Cloud Enterprise Performance Management，例如 Google APIGEE、IBM Data Power 和其他反向代理服务器。
>   要实现这一点，请通过将目标设置为云 EPM 环境的 URL（不添加 `/epmcloud` 这类上下文）来配置网关或反向代理。示例：`https://epm-idDomain.epm.dataCenterRegion.oraclecloud.com`。然后，在 login 命令中使用反向代理 URL，而非云 EPM URL。有关配置信息，请参阅网关或代理服务器的文档。
>
>   配置代理设置时，请务必将响应代码从云 EPM 传递到 EPM Automate，而不以任何方式对其进行修改，以允许 EPM Automate 正确处理响应代码，例如 200、206、400、404、500、501 等。例如，对于 IBM Datapower，将 `proxy HTTP Response` 设置为 `ON`。此外，API 网关应允许使用 HTTP 方法（GET、POST、PUT、PATCH 和 DELETE）。

在 Linux 和 UNIX 计算机上，EPM Automate 会查找以下环境变量来确定 HTTP 或 HTTPS 代理设置：

- proxyHost
- proxyPort

http 代理设置示例：

```
export proxyHost=host.example.com

export proxyPort=8000
```

https 代理设置示例：

```
export proxyHost=host.example.com

export proxyPort=8080
```

> **Note:**
>
> EPM Automate 可以使用 OAuth 2.0 身份验证协议访问云 EPM 环境（如果针对 OAuth 进行了配置）以执行命令，特别是对于自动运行命令。
>
> 在使用基本身份验证的环境中，EPM Automate 不适用于公司 SSO（身份提供程序）凭据。由于用户无法使用公司凭据登录，因此用于访问 EPM Automate 的用户帐户必须保留在身份域中。如果为您的订阅配置了 SSO，则还必须让 EPM Automate 用户能够使用其身份域凭据进行登录。请参阅《*Administering Oracle Cloud Identity Management*》中的 "启用使用身份域凭据登录"。

下载说明：《管理员入门指南》中的“下载并安装客户端”。

Java 运行时环境和 EPM Automate

在 Windows 上安装 EPM Automate 时会安装所需的 Java 运行时环境 (JRE)。但是，Linux、Unix 和 macOS X 安装程序中不包含 JRE。您必须有权访问 JRE 安装（版本 8 到版本 11）才能使用 EPM Automate。

您有权将 Oracle Java Standard Edition (SE) 与 EPM Automate 配合使用，而无需单独单独购买 Java SE 订阅。有关随 EPM Automate 授权的 Oracle JDK 的详细信息，请参阅“Oracle 支持文档 1557737.1："Support Entitlement for Java SE When Used As Part of Another Oracle Product""。

使用 OpenJDK

可以在 Linux、Unix 和 macOS X 平台上使用 OpenJDK 版本 14 或更高版本，而不使用 JRE。

可以从 https://openjdk.java.net 下载 OpenJDK，这是 Oracle 的免费、GPL 许可的生产就绪 JDK。此网站上还提供有关安装 OpenJDK 的说明。

启动 EPM Automate 会话之前，请设置 `JAVA_HOME` 环境变量以指向您的 OpenJDK 安装：

macOS X 示例（假定使用 Bash shell），使用在主目录中安装的 OpenJDK 版本 14。

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

Linux 示例（假定使用 Bash shell），使用在主目录中安装的 OpenJDK 版本 14

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

Windows 说明

默认情况下，EPM Automate 安装在 `C:/Oracle/EPM Automate` 中。

要安装 EPM Automate：

1. 从要安装 EPM Automate 的 Windows 计算机中，访问环境。
2. 在“主页”上，通过单击用户名访问**设置和操作**。
3. 单击**下载**。
4. 在“下载”页面上，单击 EPM Automate 部分中的**适用于 Windows 的下载**。
5. 将安装程序保存到计算机。
6. 右键单击安装程序 (`EPM Automate.exe`)，并选择以管理员身份运行。
7. 在**用户帐户控制**中，单击**是**。
8. 按照屏幕上的提示完成安装。

Linux/UNIX/macOS X 说明

EPM Automate 需要访问支持的 JRE（版本 8 到版本 11）的部署。环境变量 `JAVA_HOME` 必须设置为指向 JRE 安装。

要安装 EPM Automate：

1. 访问环境。
2. 在“主页”上，通过单击用户名访问**设置和操作**。
3. 单击**下载**。
4. 在“下载”页面上，单击 EPM Automate 部分中的**适用于 Linux/macOS X 的下载**。
5. 在具有读/写/执行权限的目录中保存安装程序 (`EPMAutomate.tar`)。
6. 完成以下步骤：
   a. 提取安装程序的内容
   b. 设置所需环境变量
   c. 将目录更改为从中提取安装程序内容的目录中的 `epmautomate/bin`
   d. 执行 `epmautomate.sh`：

   在以下脚本中，请记住在需要时使用实际目录路径。例如，如果 `EPMAutomate.tar` 提取到 `HOME/oracle/epmautomate/bin` 中，则将其用作 `EPMAutomate_extract_directory` 的值。

   macOS X 示例（假定使用 Bash shell），从主目录安装并运行。

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

```
cd EPMAutomate_extract_directory/epmautomate/bin
./epmautomate.sh
```

Linux 示例（假定使用 Bash shell），从主目录安装并运行。使用 JDK 版本 1.8.0_191。

```
cd ~/
tar xf directory_containing_downloaded_installer/EPMAutomate.tar
export JAVA_HOME=/opt/jdk1.8.0_191
export PATH ~/Downloads/epmautomate/bin:$PATH
cd EPMAutomate_extract_directory/epmautomate/bin
./epmautomate.sh
```

EPM Automate 命令的服务器端执行

可以使用 Groovy 直接在 Oracle Fusion Cloud Enterprise Performance Management 中运行一些 EPM Automate 命令。使用 Groovy 脚本，无需安装 EPM Automate 即可运行命令。

请注意，服务器端命令的执行与在客户端计算机上运行 Groovy 脚本来执行 EPM Automate 命令有所不同。

有关详细信息，请参阅“在不安装 EPM Automate 的情况下运行命令”。

更新 EPM Automate

EPM Automate 允许您静默升级到最新版本。

登录以启动会话后，EPM Automate 会识别当前安装的版本。如果安装的版本不是最新的可用版本，它会通知您有新版本可用。

运行 upgrade 命令静默升级 EPM Automate。

卸载 EPM Automate

> **Note:**
>
> 仅当您要从客户端计算机中删除 EPM Automate 时才使用这些步骤。如果要更新当前的 EPM Automate 版本，请使用“更新 EPM Automate”中的说明。

Windows

要卸载 EPM Automate：

1. 从 **设置**中，依次打开**应用程序**和**已安装的应用程序**
2. 在已安装的应用程序列表中找到 EPM Automate。
3. 从**溢出**菜单中，选择**卸载**。
4. 单击**卸载**。
5. 从安装目录（通常为 C:\Oracle\Epm Automate）中删除所有剩余文件

Linux/UNIX/macOS X

删除 EPM Automate 目录和 `EPMAutomate.tar` 文件。

了解 EPM Automate 加密级别

Oracle Fusion Cloud Enterprise Performance Management 将传输层安全性 (TLS) 与 SHA-2/SHA-256 加密哈希算法配合使用，以确保与 EPM Automate 的通信安全可靠。

将 OAuth 2.0 授权协议用于 OCI（第 2 代）环境

EPM Automate 可以使用 OAuth 2.0 身份验证协议访问 OCI (GEN 2) Oracle Fusion Cloud Enterprise Performance Management 环境以执行命令，特别是对于自动运行命令。

要启用 OAuth 2.0 访问，身份域管理员必须将您的应用程序注册为 Oracle Cloud Identity Services 中的公共客户端。OAuth 对应用程序强制执行；而不是在您的订阅中强制执行。

有关为您的 OCI（第 2 代）环境设置 OAuth 2.0 的详细说明，请参阅《*REST APIs*》指南中的 "Authentication with OAuth 2 - Only for OCI (Gen 2) Environments"。

> **Note:**
>
> 即使为环境启用了 OAuth，基本身份验证也有效。如果计划将来使用现有的加密密码文件，请确保不要覆盖该文件。

创建包含刷新令牌和客户端 ID 的加密密码文件

要使用 OAuth 2.0 对环境进行 EPM Automate 访问的服务管理员需要这些详细信息来创建他们的加密密码文件，然后使用该文件登录到环境中：

- 刷新令牌
  有关如何获取刷新令牌的详细说明，请参阅《*REST APIs*》指南中的 "Authentication with OAuth 2 - Only for OCI (Gen 2) Environments"。
- 客户端 ID
  身份域管理员 为 OAuth 配置应用程序时生成客户端 ID。它在应用程序的“配置”选项卡上可见，位于**常规信息**下。

要为 OAuth 身份验证创建加密密码文件：

1. 启动 EPM Automate 会话。
2. 执行类似下面的命令：
   `epmautomate encrypt` *`REFRESH_TOKEN ENCRYPTION_KEY PASSWORD_FILE`* `ClientID=`*`CLIENT_ID`*，其中，*`REFRESH_TOKEN`* 是来自安全存储的已解密刷新令牌，*`ENCRYPTION_KEY`* 是用于密码加密的私钥，*`PASSWORD_FILE`* 是存储已加密刷新令牌的文件的名称和位置。密码文件必须使用 `.epw` 扩展名。

   有关详细说明，请参阅 encrypt。
3. 使用新生成的密码文件来使用 OAuth 登录。对于自动脚本执行，请务必更新脚本以指向新生成的密码文件。

2

命令参考

- 关于运行 EPM Automate 命令
- 运行 EPM Automate
- 命令概览
- EPM Automate 命令
- 退出代码

某些 EPM Automate 命令适用于所有业务流程，而某些适用于一组业务流程。除非另外指明，否则适用于特定业务流程（例如 Planning）的命令对其他业务流程（例如 Financial Consolidation and Close）不起作用。尝试对业务流程执行它不支持的命令会导致错误。有关适用于每个业务流程的命令列表，请参阅“特定于每个云 EPM 服务的命令”。

关于运行 EPM Automate 命令

所有 Oracle Fusion Cloud Enterprise Performance Management 服务都使用 EPM Automate 命令来远程管理环境。

- 先决条件
- 默认文件位置
- 启用传输层安全协议 1.2
- 使用 EPM Automate 命令
- 为一个参数指定多个值
- 日常维护期间的行为

先决条件

本节列出了使用 EPM Automate（例如，使用 Oracle Fusion Cloud Enterprise Performance Management 凭据和环境中的默认文件位置）的先决条件。

常规

所有云 EPM 用户均可使用其身份域凭据通过 EPM Automate 连接到环境。分配给用户的预定义角色和应用程序角色决定用户可以执行的命令。

- 此外，若要在身份域中运行命令以添加或删除用户，则需要身份域管理员角色。
- 执行命令所需的任何文件都必须存在于环境中。您可以使用 uploadFile 命令上传文件。

  有关每个服务所用默认文件位置的信息，请参阅“默认文件位置”。
- 命令中的文件扩展名用法：
  - 指定完整文件名，包括文件扩展名（例如，data.csv）来运行执行文件操作的命令。文件操作命令示例包括 `deletefile`、`listfiles`、`uploadfile`。
  - 不使用文件扩展名来运行执行迁移操作的命令。迁移操作需要指定快照的名称。

- 包含空格字符的参数值（例如注释、位置名称和文件夹路径）必须用引号括起来。

Planning

- 作业
  下节中讨论的许多命令都需要使用作业。作业是导入或导出数据这类操作，可以立即开始也可以调度在以后执行；例如，导入或导出数据以及刷新数据库。

  使用作业控制台，您必须创建相应的作业才能执行以下操作。有关在 Planning 中创建作业的详细说明，请参阅《管理 *Planning*》中的“管理作业”。

  - 将数据导入应用程序中
  - 从应用程序导出数据
  - 将元数据导入应用程序中
  - 从应用程序导出元数据
  - 将数据从一个块存储数据库复制到一个聚合存储数据库或者从一个块存储数据库复制到另一个块存储数据库
- 业务规则
  您要执行的业务规则必须存在于应用程序中。

  您可以使用 Calculation Manager 创建业务规则，这些业务规则随后将部署到应用程序中。请参阅使用 *Calculation Manager* 进行设计。

数据管理

- 数据规则
  数据加载规则定义数据管理如何从文件中加载数据。您必须具有预定义的数据加载规则，才能使用 EPM Automate 加载数据。
- 批处理
  您可以使用数据管理中定义的批处理来加载数据。使用批处理，用户可以合并一个批处理中的多个加载规则并以串行或并行模式执行这些规则。

默认文件位置

默认上传位置

默认情况下，上传到 Oracle Fusion Cloud Enterprise Performance Management 的所有文件都存储在迁移可访问的默认位置。

您必须将要由迁移处理的文件（例如，要导入到服务的快照）上传到默认位置。

收件箱和发件箱

收件箱和发件箱的位置可能因云 EPM 业务流程而异。您可以使用收件箱上传要导入的文件或者要由 Profitability and Cost Management 以外的业务流程处理的文件。数据管理可以处理收件箱中或收件箱的目录中的文件。

通常，云 EPM 将通过业务流程生成的文件（例如，数据或元数据导出文件）存储在发件箱中。

- EPM Automate 将文件上传到的收件箱和存储要下载的文件的发件箱可供以下应用程序访问。如果您计划使用这些应用程序的本地进程处理文件，必须将这些文件上传到此位置。您也可以将文件上传到发件箱。
  - Planning
  - Planning 模块

  - Account Reconciliation
  - Financial Consolidation and Close
  - Tax Reporting
  - Narrative Reporting
  - Enterprise Profitability and Cost Management

  可以使用收件箱/发件箱资源管理器浏览在默认位置存储的文件。您使用 EPM Automate 创建的应用程序快照不会列在收件箱/发件箱资源管理器中；可以通过迁移的“快照”选项卡进行查看。

- 要使用 Profitability and Cost Management 流程处理的文件必须上传到 `profitinbox` 中。也可以将文件上传到 profitoutbox。Profitability and Cost Management 流程导出的文件将存储在 `profitinbox` 中。可使用文件资源管理器浏览这些文件。
- 要使用数据管理处理的文件必须位于收件箱中或收件箱内的某个文件夹中。默认情况下，使用数据管理导出的文件存储在发件箱中，而数据管理报表输出存储在数据管理的 `outbox/report` 文件夹中。可使用数据管理文件浏览器来浏览这些文件。
- Oracle Fusion Cloud Enterprise Data Management 使用默认位置来存储已上传、复制或下载的导入和导出文件。可以使用 `ListFiles` 命令来查看默认位置中的文件。

日志文件

每次 EPM Automate 命令执行都会生成调试文件，如果命令成功，该文件会自动删除。如果在命令执行期间发生错误，则将在运行 EPM Automate 的目录中维护失败命令的调试文件。默认情况下，这是 `Oracle/epm automate/bin` 目录 (Windows) 或 `home/user/epmautomate/bin` (Linux/UNIX)。

EPM Automate 调试文件使用以下命名惯例：

`commandname_date_timestamp.log`。例如，如果在 2020 年 11 月 23 日 09:28:02 运行失败的 `listfiles` 命令，则调试文件名为 `listfiles_23_11_2020_09_28_02.log`。

无法禁止创建失败命令的调试文件。但是，您可以通过将 `-d` 以及调试文件名和出错及输出流 (`-d >> c:\logs\LOG_FILE_NAME.log 2>&1`) 附加到命令的结尾，将调试信息和命令输出写入到不同目录的文件，如以下 Windows 示例所示：

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

启用传输层安全协议 1.2

EPM Automate 必须安装在支持传输层安全 (TLS) 协议 1.2 或更高版本的操作系统上。

为了确保身份验证和数据加密获得最高级别的安全性，EPM Automate 仅支持 TLS 1.2。如果运行 EPM Automate 的计算机上未启用 TLS 1.2，将显示 `EPMAT-7: Unable to connect.Unsupported Protocol: HTTPS` 错误。要解决此错误，请与 IT 管理员协作来启用 TLS 1.2。

启用 TLS 1.2 的过程与操作系统有关。请使用以下信息源；对于其他受支持的操作系统，可以使用类似的 Web 资源：

- 将更新，以启用 TLS 1.1 和 TLS 1.2 作为默认 WinHTTP 在 Windows 中的安全协议，可获取有关为 Windows 计算机启用 TLS 1.2 的信息。
- 强化 TLS 配置，以获取有关在 OpenSSL for Red Hat Enterprise Linux 中启用 TLS 1.2 的信息。

使用 EPM Automate 命令

命令参数的顺序

命令的所有必需参数均须按照命令用法中确定的顺序传递。必需的参数及其值在可选参数的前面，可选参数可按任意顺序传递。可选参数不是位置参数。

例如，请考虑 `login` 命令的以下用法：

```
epmautomate login USERNAME PASSWORD CLOUD_EPM_BASE_URL
[ProxyServerUserName=PROXY_USERNAME] [ProxyServerPassword=PROXY_PASSWORD]
[ProxyServerDomain=PROXY_DOMAIN]
```

此命令有三个必需的参数：`USERNAME`、`PASSWORD` 和 `CLOUD_EPM_BASE_URL`，它们应该按照用法中确定的顺序显示。如果不按此顺序显示，此命令将返回错误。可选参数 `ProxyServerUserName`、`ProxyServerPassword` 和 `ProxyServerDomain` 及其值可按任意顺序指定。

EPM Automate 命令是否区分大小写？

EPM Automate 命令不区分大小写。键入命令名称的方式对命令执行没有影响。例如，对于 `addUsers` 命令，可以键入为 `addusers`、`ADDUSERS` 或 `AdDuSeRs`。

EPM Automate 命令参数是否区分大小写？

EPM Automate 命令参数不区分大小写。键入命令参数名称的方式对命令执行没有影响。例如，您可以将 `FileName` 参数键入为 `filename`、`fileName` 或 `fIlEnAmE`，不会影响命令执行。

为一个参数指定多个值

有些 EPM Automate 命令接受多个以逗号分隔的参数值；例如，Planning 应用程序的业务规则、规则集和模板中类型成员的运行时提示。

要在 EPM Automate 命令中为名为 `Entities` 的运行时提示的成员类型设置多个成员，请在执行 `runbusinessrule` 命令时按下例所示使用逗号 (,)。

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

包含空格和逗号等特殊字符的成员必须放在双引号中，并使用 \（反斜杠）对双引号转义，如下例所示：

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

日常维护期间的行为

在对环境进行日常维护时，不要运行 EPM Automate 命令。

日常维护期间不允许执行用户活动。在进行日常维护时，尝试直接或使用脚本运行 EPM Automate 命令将显示以下错误：

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

运行 EPM Automate

您可以通过 EPM Automate 使用 Oracle Fusion Cloud Enterprise Performance Management 凭据登录。不能使用自己的 SSO 凭据登录。

所有云 EPM 用户均可使用其身份域凭据通过 EPM Automate 连接到环境。分配给用户的预定义角色和应用程序角色决定用户可以执行的命令。

此外，只有服务管理员可以运行某些命令，而运行某些命令可能还需要身份域管理员角色。

生成调试日志文件

Oracle 技术支持将要求您提供会话的调试日志文件，以便解决您在运行 EPM Automate 时遇到的问题。它支持使用 `-d` 选项以生成调试消息，之后可以使用 `>` 指令将这些消息重定向到一个文件。您可以为一个命令或者包含多个命令的批处理执行文件或脚本创建一个调试文件。

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

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

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

Windows

运行 EPM Automate 之前，确保您可以从运行 EPM Automate 的计算机中访问您的环境。

EPM Automate 会在当前目录中创建一个 `.prefs` 文件（其中包含用户信息）和日志文件。在 Windows 计算机上，`.prefs` 文件的内容仅对创建它的用户以及 Windows 管理员可见。在 Linux、UNIX 和 macOSX 环境中，`.prefs` 文件使用权限 600 生成，该权限仅允许所有者对此文件进行读取和写入。

如果您对执行 EPM Automate 的 Windows 目录没有写入权限，则 EPM Automate 将在 Windows 环境中显示 `FileNotFoundException: .prefs (Access is denied)` 错误。要解决此错误，请确保当前用户的 Windows 帐户对从中运行 EPM Automate 的目录具有读/写访问权限。此外，该用户还必须对从中访问（例如，在运行 `uploadFile` 命令时）或写入（例如，在运行 `downloadFile` 命令）文件的任何其他目录具有适当的访问权限。

> **注：**
>
> 不能从名称中包含 `&` 的文件夹运行 EPM Automate；例如 `C:\Oracle\A&B`。

要在 Windows 客户端上运行 EPM Automate：

1. 依次单击**开始**、**所有程序**、**EPM Automate** 和**启动 EPM Automate**。此时将显示 EPM Automate 命令提示符。
2. **可选：**导航到您要在其中执行操作的目录。
3. **可选：**生成密码加密文件。您可使用密码加密文件传递加密密码，以启动会话。

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

4. 以服务管理员身份启动一个会话。使用类似下面的命令：

- 使用未加密密码：

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

- 使用加密密码：

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

5. 输入命令以执行您要完成的任务。

   有关命令执行状态的信息，请参阅“退出代码”。

6. 从环境中注销。使用以下命令：

```
epmautomate logout
```

Linux

> **注：**
>
> 确保在 `.profile` 文件的 `PATH` 变量中设置了 *`JAVA_HOME`* 或者将其设置为 shell 环境变量。需要支持的 JRE（版本 8 到 11）。

要在 Linux 客户端上运行 EPM Automate：

1. 打开一个终端窗口，并导航到安装 EPM Automate 的目录。
2. **可选：**生成密码加密文件。您使用密码加密文件传递加密密码（而不是未加密密码）以启动会话。

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

3. 以服务管理员身份启动一个会话。使用类似下面的命令：
   - 使用未加密密码：

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

   - 使用加密密码：

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

4. 输入命令以执行您要完成的任务。

   有关命令执行状态的信息，请参阅“退出代码”。

5. 从环境中注销。使用以下命令：

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

运行 EPM Automate 的多个实例

您可从同一个目录针对一个环境运行 EPM Automate 的多个实例。类似地，您也可以从同一个目录或不同目录针对多个环境运行多个实例。
例如，您可能需要在 `https://cloudpln.pbcs.us1.oraclecloud.com` 和 `https://testcloudpln.pbcs.us1.oraclecloud.com` 中同时刷新 Planning 应用程序多维数据集。在此方案中，您有两种选择：

- 从同一个目录运行 EPM Automate 的两个实例，分别刷新不同环境中的应用程序多维数据集
- 从两个单独的目录分别执行 EPM Automate 以连接到环境，然后刷新应用程序多维数据集

在这两种方案中，每一个实例都独立运行；从一个实例注销不会使您从另一个实例注销。即使从另一个实例注销，使用 EPM Automate 启动的活动也将继续在环境中运行直至完成。

本节包含可用于创建两个 EPM Automate 会话以执行任务的 Windows 和 Unix/Linux 示例脚本（`caller` 和 `multisession`）。要同时运行多个会话，必须在 `caller` 脚本中添加以下连接信息，该脚本会调用 `multisession` 脚本来运行 `login`、`uploadfile`、`listfiles` 和 `logout` 命令。可以修改 `multisession` 脚本来执行这些任务之外的其他任务。确保这两个脚本存储在同一目录中。

- EPM Automate 使用环境变量 `EPM_SID` 区分多个会话。必须在 caller 脚本中将此变量设置为每个会话的唯一值。在示例脚本中，将其设置为唯一值，如下所示：
  - 在 `caller.BAT` 中，`EPM_SID` 设置为 `!RANDOM!`，这会向其分配系统生成的唯一数字。此数字同时用于为每个会话生成日志文件。如果要跟踪每个会话的日志文件，可以为其指定唯一的数字而不是 `!RANDOM!`。
  - 在 `caller.sh` 中，`EPM_SID` 设置为唯一的进程 ID。如果要跟踪每个会话的日志文件，可以指定唯一的 `EPM_SID`，方法是在 `multisession` 脚本中修改 `export EPM_SID=$$` 语句以使用传入的值，然后在 `caller` 脚本中针对每个会话为此参数传递唯一值，例如，在 `caller.sh` 中指定 `EPM SID` 的值，如下所示：

    ```
    $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`：服务管理员的登录 ID
- `PASSWORD`：服务管理员的密码
- `URL`：环境的连接 URL

Windows 脚本示例

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
```

Bourne Shell 脚本示例

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
```

命令概览

下面按字母顺序列出了所有 EPM Automate 命令。

表 2-1　所有 EPM Automate 命令

命令名称	PLN、SWP、SP、FF	FCC	TR	PCM	EPCM	AR	EDM	NR
addUsers	✓	✓	✓	✓	✓	✓	✓	✓
addUsersToGroup	✓	✓	✓	✓	✓	✓	✓	✓
addUsersToTeam		✓	✓			✓		
addUserToGroups	✓	✓	✓	✓	✓	✓	✓	✓
applicationAdminMode	✓	✓	✓		✓			
applyDataGrants				✓				
archiveTmTransactions						✓		
assignRole	✓	✓	✓	✓	✓	✓	✓	✓
autoPredict* 参阅脚注	✓							
calculateModel					✓			
clearCube	✓				✓			
clearDataByPointOfView					✓			
clearDataByProfile		✓	✓					
clearPOV				✓				
cloneEnvironment	✓	✓	✓	✓	✓	✓	✓	✓
compactCube	✓				✓			
copyDataByPointOfView					✓			
copyDataByProfile		✓	✓					
copyFileFromInstance	✓	✓	✓	✓	✓	✓	✓	✓
copyFromObjectStorage	✓	✓	✓	✓	✓	✓	✓	✓
copyFromSFTP	✓	✓	✓	✓	✓	✓	✓	✓
copyOwnershipDataToNextYear		✓	✓					
copyPOV				✓				
copySnapshotFromInstance	✓	✓	✓	✓	✓	✓	✓	
copyToObjectStorage	✓	✓	✓	✓	✓	✓	✓	✓
copyToSFTP	✓	✓	✓	✓	✓	✓	✓	✓
createGroups	✓	✓	✓	✓	✓	✓	✓	✓
createNRSnapshot								✓
createReconciliations						✓		

表 2-1 （续） 所有 EPM Automate 命令

命令名称	PLN、SWP、SP、FF	FCC	TR	PCM	EPCM	AR	EDM	NR
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				✓				
exportSnapshot	✓	✓	✓	✓	✓	✓	✓	
exportTemplate				✓				

表 2-1 （续） 所有 EPM Automate 命令

命令名称	PLN、SWP、SP、FF	FCC	TR	PCM	EPCM	AR	EDM	NR
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	✓	✓	✓	✓	✓	✓	✓	
importSupplementalCollectionData		✓	✓					
importSupplementalData		✓	✓					

表 2-1 （续） 所有 EPM Automate 命令

命令名称	PLN、SWP、SP、FF	FCC	TR	PCM	EPCM	AR	EDM	NR
importTemplate				✓				
importTMAttributeValues						✓		
importValidIntersections	✓	✓	✓		✓			
invalidLoginReport	✓	✓	✓	✓	✓	✓	✓	✓
listBackups	✓	✓	✓	✓	✓	✓	✓	✓
listFiles	✓	✓	✓	✓	✓	✓	✓	✓
loadData				✓				
loadDimData				✓				
loadDimensionViewpoint							✓	
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						✓		
runDailyMaintenance	✓	✓	✓	✓	✓	✓	✓	✓

表 2-1 （续） 所有 EPM Automate 命令

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

- * 仅当在应用程序中启用了混合 Oracle Essbas 多维数据集时，才支持此命令。战略性人员规划和销售规划不支持混合 Essbase。自由形式不支持此命令。

- ** 自由形式不支持此命令。

缩写词

- PLN：Planning（包括 Planning 模块）
- FF：自由形式
- 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 Fusion Cloud Enterprise Data Management
- NR：Narrative Reporting

EPM Automate 命令

本节详述了每个 EPM Automate 命令。每个命令的可用信息包括可使用该命令的服务、命令用法和示例。

addUsers

使用上传到环境中的 ANSI 或 UTF-8 编码的逗号分隔值 (CSV) 文件在身份域中创建一批用户。同时向新用户告知其用户名和临时密码。

服务管理员使用 uploadFile 命令将文件上传到环境中。CSV 文件中的所有列都是必填的。此命令验证某个定义的每一列中的值，并显示标识每个缺少或无效值的错误消息。CSV 文件格式如下：

```
First Name,Last Name,Email,User Login
Jane,Doe,jane.doe@example.com,jdoe
John,Doe,john.doe@example.com,john.doe@example.com
```

有关 CSV 文件格式的详细说明，请参阅《*Getting Started with Oracle Cloud*》中的 "Importing a Batch of User Accounts"。

导入文件中指定的 `User Login` 的值不区分大小写。例如，值 `John.doe@example.com` 被视为与 `John.Doe@example.com` 或其任何大小写变体相同。

如果 CSV 文件中的用户定义与身份域中存在的用户帐户相匹配，则不会对现有的用户帐户执行任何更改。此命令仅为其帐户信息已包含在此文件中的新用户创建帐户。由于用户帐户在身份域支持的所有环境中是公用的，因此新用户会提供给共享身份域的所有环境。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost

Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

身份域管理员和任何预定义角色

用法

`epmautomate addUsers FILE_NAME [userPassword=PASSWORD] [resetPassword=true|false]`，其中：

- *FILE_NAME* 是包含用户信息的 CSV 文件的名称。包含多字节字符的输入文件必须使用 UTF-8 字符编码。
- `userPassword`（可选）指示在身份域中创建的所有新用户的默认密码。如果指定，此密码必须符合身份域密码的最低要求。如果未指定此参数，则会为每个用户分配一个唯一的临时密码。
  如果指定，则 `userPassword` 参数用作 CSV 文件中指定的所有用户的密码。如果仅仅是为了进行测试而创建用户，则可以为所有用户分配同一密码。如果要创建 Oracle Fusion Cloud Enterprise Performance Management 真实用户而且希望为每个用户分配一个特定密码，请使用此命令而不为 `userPassword` 可选参数指定值。
- `resetPassword`（可选）指示新用户是否必须在首次登录时更改密码。默认值为 `true`。除非此参数设置为 `false`，否则将强制新用户在首次登录时更改密码。
  如果 `resetPassword` 设置为 `true`，则此命令向每个新用户发送一封电子邮件，该电子邮件中包含有关新用户帐户的详细信息（用户名和密码）。如果 `resetPassword` 设置为 `false`，则不会发送电子邮件。如果您将 `resetPassword` 设置为 false，则必须指定 `userPassword`。否则，将为每个用户分配一个唯一的临时密码，但是由于未发送电子邮件，用户将不知道密码，因此将无法登录。

示例

- 在身份域中添加具有相同密码的测试用户，且不要求他们更改密码：
  ```
  epmautomate addUsers user_file.CSV userPassword=Example@Pwd12
  resetPassword=false
  ```
- 向身份域中添加具有临时密码的用户，并要求他们更改临时密码：
  ```
  epmautomate addUsers user_file.CSV
  ```

addUsersToGroup

使用上传到环境中的 ANSI 或 UTF-8 编码的 CSV 文件将一批用户添加到访问控制中的现有组。

服务管理员使用 uploadFile 命令将文件上传到环境中。用户登录值不区分大小写。文件格式如下：

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

> **注：**
>
> 仅当下面两个条件都满足时才能将用户添加到组：
>
> - 文件中包括的用户登录值存在于为环境提供服务的身份域中。用户登录值不区分大小写。
> - 将为用户分配身份域中的预定义角色

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Enterprise Profitability and Cost Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate addUsersToGroup FILE_NAME GROUP_NAME`，其中：

- *FILE_NAME* 是 CSV 文件的名称，该文件包含要分配给访问控制。中某个组的用户的登录名
- *GROUP_NAME* 是访问控制。中存在的组的名称此值不区分大小写。

示例

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

addUsersToTeam

将 CSV 文件中列出的 Oracle Fusion Cloud Enterprise Performance Management 用户添加到现有团队。

如果 CSV 文件中包含的某个用户已是团队成员，此命令将忽略该用户。此文件中的值不区分大小写。服务管理员使用 uploadFile 命令将此文件加载到环境中。CSV 文件格式如下：

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

> **注：**
>
> 默认情况下，指定主用户执行分配给团队的任务。

适用于

Financial Consolidation and Close、Tax Reporting 和 Account Reconciliation。

所需角色

- 服务管理员
- 任何预定义角色和“团队 - 管理”应用程序角色
- 任何预定义角色和“用户 - 管理”应用程序角色

用法

`epmautomate addUsersToTeam FILE TEAM_NAME`，其中：

- `FILE` 确定 UTF-8 格式的 CSV 文件，其中列出要添加到团队的用户的登录 ID。运行此命令前，请使用 uploadFile 命令将文件上传到环境。
- `TEAM_NAME` 确定在访问控制中定义的团队名称。此值不区分大小写。

示例

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

addUserToGroups

添加用户作为 ANSI 或 UTF-8 编码 CSV 文件中所标识的访问控制组的成员。

服务管理员使用 uploadFile 命令将文件上传到环境中。文件格式如下：

```
Group Name
Group1
Group2
```

组名的值不区分大小写。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate addUserToGroups FILE_NAME User_Login`，其中：

- `FILE_NAME` 是 CSV 文件的名称，该文件包含要向其中分配该用户的访问控制组名称

- `User_Login` 是要分配到访问控制组的 Oracle Fusion Cloud Enterprise Performance Management 用户的登录 ID。此用户登录 ID 不区分大小写，必须存在于为环境提供服务的身份域中，且必须已分配有预定义角色。

示例

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

applicationAdminMode

将应用程序置于管理模式，以仅限服务管理员可以访问该应用程序。

要在服务管理员正在执行管理操作时防止其他用户使用该应用程序，该命令十分有用。应用程序将一直处于管理模式，直至您重新做出更改后才可供所有用户访问。

> **注：**
>
> 此命令已废弃，但未从 EPM Automate 中删除。Oracle 建议您改用 setApplicationAdminMode 命令。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate applicationAdminMode` *`VALUE`*，其中 `VALUE` 指定是否将应用程序置于管理模式。可接受的值包括：

- `true` 表示将应用程序置于管理模式
- `false` 表示将应用程序恢复为正常模式，以供所有用户访问

示例

- 将应用程序置于管理模式： `epmautomate applicationAdminMode true`
- 将应用程序恢复为正常运行： `epmautomate applicationAdminMode false`

applyDataGrants

刷新控制 Oracle Essbase 数据切片访问的数据授权，使这些数据授权与 Profitability and Cost Management 应用程序中定义的数据授权相匹配。

您在 Profitability and Cost Management 应用程序中设置的用户和组级别数据授权将在 Essbase 中自动同步。如果您怀疑应用程序中的数据授权与 Essbase 中的筛选器存在不一致，可使用此命令同步对 Essbase 数据的访问权限。

完成此操作所需的时间取决于应用程序的大小。确保在下次维护期间备份应用程序之前完成数据授权刷新操作。因为在此操作进行期间不应使用应用程序，Oracle 建议您将此操作安排在用户不使用应用程序的时段进行。

适用于

Profitability and Cost Management

所需角色

服务管理员、超级用户

用法

`epmautomate applyDataGrants APPLICATION_NAME`，其中 *APPLICATION_NAME* 是要为其重新创建数据授权的 Profitability and Cost Management 应用程序的名称。

示例

```
epmautomate applyDataGrants BksML12
```

archiveTmTransactions

存档等于或超过指定天数的匹配事务（包括支持和调整详细信息）。匹配事务记录在 ZIP 文件中。

使用此命令可以根据组织的事务保留策略存档并清除旧的匹配事务，使 Account Reconciliation 应用程序大小保持最佳。

适用于

Account Reconciliation

所需角色

服务管理员、超级用户、用户、查看者
具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

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

- `matchType` 是应存档匹配事务的匹配类型的标识符 (`TextID`)。
- `age` 标识自匹配事务以来的天数。将存档超过或等于此值的匹配事务。
- `filterOperator`（可选）是以下筛选条件之一，用于标识包含要存档的匹配事务的帐户。此值与 `filterValue` 相结合，用于标识应存档匹配事务的帐户：
  - `equals`
  - `not_equals`
  - `starts_with`
  - `ends_with`
  - `contains`
  - `not_contains`
- `filterValue`（可选）是用于标识要存档的事务的筛选器值。如果 `filterOperator` 为 `equals` 或 `not_equals`，则可以使用空格分隔的列表指定多个值；例如

filterValue=101-120 filterValue=102-202。如果指定多个值，将选择与任何筛选器运算符和筛选器值组合匹配的帐户中的事务进行存档。

> **Note:**
>
> 如果未指定 filterOperator 和 filterValue，则将存档所有帐户中指定 matchType 的超过或等于 age 的所有匹配事务。

- logFilename（可选）是用于记录命令活动相关信息的日志文件的名称。如果未指定文件名，则将自动生成名为 Archive_Transactions_*matchType*_*JOBID*.log 的日志文件。
- filename（可选）是应包含存档事务的 .ZIP 文件的名称。如果未指定此项，则默认情况下，命令将创建 Archived_Transactions_*matchType*_*JOBID*.zip。使用 downloadFile 命令将此文件下载到本地计算机。

> **Note:**
>
> 此命令使用您指定的参数运行“存档 TM 事务”作业。将在命令输出中返回作业 ID，以方便您在 purgeArchivedTmTransactions 命令中使用它。您可以从作业控制台监视作业。

示例

- 不使用筛选器而是使用自定义日志和 .ZIP 文件名来存档旧的匹配事务：
  ```
  epmautomate archiveTmTransactions cashrecon 180 logFile=tmlogs.log
  filename=trans.zip
  ```
- 使用筛选器来存档旧的匹配事务：
  - ```
    epmautomate archiveTmTransactions cashrecon 180 filterOperator=equals
    filterValue=101-120 FilterValue=102-202
    ```
  - ```
    epmautomate archiveTmTransactions cashrecon 180 filterOperator=contains
    filterValue=11
    ```

assignRole

为 ANSI 或 UTF-8 编码 CSV 文件中包含其登录 ID 的用户（包括运行此命令的用户）分配角色。使用此命令将用户分配给预定义角色或应用程序角色。

使用此命令之前，服务管理员使用 uploadFile 命令将输入文件上传到环境中。文件格式如下：

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

请参阅《*Getting Started with Oracle Cloud*》中的 "Assigning One Role to Many Users"。

> **注：**
>
> - 文件中包含的用户登录值不区分大小写。
> - 使用双引号将包含空白字符的角色名称括起来。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

要分配预定义角色：

- 要分配预定义角色：
  - 服务管理员
  - 身份域管理员和任何预定义角色
- 要分配应用程序角色：
  - 服务管理员
  - 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate assignRole` *`FILE_NAME ROLE`*，其中：

- *`FILE_NAME`* 是包含用户登录 ID 的 CSV 文件的名称。以小写形式指定 CSV 扩展名。
- `ROLE` 是下列值之一。此值不区分大小写：
  - 若要为用户分配预定义的身份域角色，则 `ROLE` 应标识适用于服务的预定义角色。请参阅《管理员入门指南》中的“了解预定义角色”。
    有关这些角色的说明，请参阅《管理访问控制》中的“在应用程序级别管理角色分配”
  - 如果您为用户分配应用程序角色，`ROLE` 应标识属于当前环境中应用程序的角色。访问控制的**角色**选项卡中列出了应用程序角色。有关每个业务流程的应用程序角色说明，请参阅《管理 *Oracle Enterprise Performance Management Cloud* 访问控制》中的以下主题：
    - Account Reconciliation
    - Enterprise Profitability and Cost Management
    - Planning、自由形式、Financial Consolidation and Close 和 Tax Reporting
    - Profitability and Cost Management
    - Oracle Enterprise Data Management
    - Narrative Reporting

示例

- 为用户分配预定义的身份域角色：
  ```
  epmautomate assignRole admin_role_file.csv "Service Administrator"
  ```
- 为用户分配应用程序角色：
  ```
  epmautomate assignRole example_file.csv "Task List Access Manager"
  ```

autoPredict

根据 Planning 或 Planning 模块中的现有自动预测定义生成对未来绩效的预测。

此命令启动一个作业，该作业使用在应用程序中所指定自动预测定义中标识的每个成员的历史数据。有关使用自动预测功能的应用程序以及设置预测的详细信息，请参阅《管理 *Planning*》中的“设置预测以使用自动预测来自动运行”。

适用于

Planning 和 Planning 模块（如果在应用程序中启用了混合 Oracle Essbase 多维数据集）。

所需角色

服务管理员

用法

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

- *`PREDICTION_DEFINITION`* 是应用程序中可用的自动预测定义的名称。
- `forceRun`（可选）指定如果初始运行后基础定义没有更改，是否运行预测。默认值为 `false` 将此参数的值设置为 `true` 以运行自动预测作业，即使作业定义中没有更改。使用默认值 (`false`) 在作业第一次执行时运行一次预测。
- `paginatedDim`（可选）指定维，用于通过在单独的线程中并行运行测试来加快自动预测作业的速度。为了使这些并行线程高效运行，请指定将使数据针对每个预测线程平均分布的维。

示例

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

calculateModel

在 Enterprise Profitability and Cost Management 应用程序中运行计算过程。

适用于

Enterprise Profitability and Cost Management

所需角色

服务管理员

用法

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

- *POV_NAME* 是要计算的数据 POV 的名称。要计算多个 POV，请列出用逗号分隔符分隔的 POV 名称。请勿使用任何其他分隔符分隔 POV 名称。如果成员名称中存在空格，请用双引号将 POV 名称列表括起来。
- *MODEL_NAME* 是要计算的模型的名称。如果模型名称包含空格，请用双引号将名称括起来。
- *EXECUTION_TYPE* 是下列值之一，用于标识规则执行类型。
  - ALL_RULES 表示使用所有规则来计算 POV。
    如果指定此值，请不要指定与规则子集或单个规则相关的运行时参数，例如 rulesetSeqNumStart、rulesetSeqNumEnd 和 ruleName。
  - RULESET_SUBSET 表示使用规则集的子集来计算 POV。
    如果使用此值，则必须指定 rulesetSeqNumStart 和 rulesetSeqNumEnd 值作为运行时参数。
  - SINGLE_RULE 表示运行特定的规则来计算 POV。
    如果使用此值，则只能指定 ruleName 作为运行时参数。
  - RUN_FROM_RULE 表示从特定规则开始对 POV 运行计算。
    如果使用此值，则只能指定 ruleName 作为运行时参数。
  - STOP_AFTER_RULE 表示在特定规则完成计算后停止计算 POV。
    如果使用此值，则只能指定 ruleName 作为运行时参数。
- povDelimiter（可选）是 POV 值中使用的分隔符。默认分隔符为 ::（双冒号）。分隔符必须用双引号括起来。仅支持以下分隔符：
  - _（下划线）
  - #（井号）
  - &（和号）
  - ~（波形符）
  - %（百分比）
  - ;（分号）
  - :（冒号）
  - -（短划线）
- optimizeForReporting=true|false（可选）指定运行计算时是否针对报告进行优化。默认值为 false。
  将此值设置为 false 可通过跳过聚合创建步骤来节省处理时间；例如，在运行单个规则或连续系列的 POV 时。运行多个并发的计算作业时，为所有作业设置 optimizeForReporting=true，因此只有最后一个要完成的作业将执行聚合，这样可以避免冗余处理并防止作业的运行速度下降。
- captureDebugScripts=true|false（可选）标识是否在收件箱中生成调试脚本。Oracle 可能需要这些脚本来解决计算问题。默认值为 false。
- comment="COMMENT"（可选）在双引号中指定有关进程的注释。
- PARAMETER=*VALUE*（可选）指明用于运行计算的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：
  - rulesetSeqNumStart 指定要运行的规则集中第一条规则的序号。仅在使用 EXECUTION_TYPE=RULESET_SUBSET 时有效。
  - rulesetSeqNumEnd 指定要运行的规则集中最后一个规则的序号。仅在使用 EXECUTION_TYPE=RULESET_SUBSET 时有效。

- `ruleName` 指定要运行的规则的名称。如果值包含空格字符，请用双引号将值括起来。仅当 `EXECUTION_TYPE` 的值设置为 `SINGLE_RULE`、`RUN_FROM_RULE` 或 `STOP_AFTER_RULE` 时有效。
- `clearCalculatedData=true|false` 指定是否清除现有计算。默认值为 `false`。
- `executeCalculations=true|false` 指定是否运行计算。默认值为 `false`。

> **Note:**
>
> 参数值（`true` 和 `false`）必须全部小写。

示例

- 运行所有规则以计算单个 POV：
```
epmautomate calculateModel FY22::Jan::Actual::Working "10 Actuals Allocation
Process" ALL_RULES clearCalculatedData=true executeCalculations=true
optimizeForReporting=true comment="Running all rules to calculate a POV"
```
- 还使用自定义 POV 分隔符，运行所有规则以计算多个 POV：
```
epmautomate calculateModel
"FY22_Jan_Actual_Working,FY22_Feb_Actual_Working,FY22_Mar_Actual_Working" "10
Actuals Allocation Process" ALL_RULES clearCalculatedData=true
executeCalculations=true optimizeForReporting=true captureDebugScripts=true
comment="Runing calculation for multiple POVs" povDelimiter="_"
```
- 运行一组规则集以计算 POV：
```
epmautomate calculateModel FY22::Jan::Actual::Working "10 Actuals Allocation
Process" RULESET_SUBSET rulesetSeqNumStart=10 rulesetSeqNumEnd=20
clearCalculatedData=true executeCalculations=true comment="Running a subset of
rule sets"
```
- 运行特定规则以计算 POV：
```
epmautomate calculateModel FY22::Jan::Actual::Working "10 Actuals Allocation
Process" SINGLE_RULE ruleName="Rent and Utilities Reassignment"
clearCalculatedData=true executeCalculations=true comment="Running a specific
rule"
```

clearCube

使用在类型为 `clear cube` 的作业中指定的设置从输入和报表多维数据集中删除特定数据。

此命令不会删除应用程序关系表中的应用程序定义。请参阅《管理 *Planning*》中的“清除多维数据集”。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate clearCube` *`JOB_NAME`*，其中：*`JOB_NAME`* 是在应用程序中定义的作业名称。

示例

```
epmautomate clearCube ClearPlan1
```

clearDataByPointOfView

清除 Enterprise Profitability and Cost Management 多维数据集的特定 POV 的数据。

适用于

Enterprise Profitability and Cost Management

所需角色

服务管理员

用法

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

- *`POV_NAME`* 是应用程序中的 POV 的名称。
- `cubeName`（可选）是要从中清除数据的多维数据集的名称。默认值为 `PCM_CLC`。
- `PARAMETER=`*`VALUE`* 表示可选的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：
  - `povDelimiter` 是 POV 值中使用的分隔符。默认值为 `::`（双冒号）。此值必须用双引号括起来。示例：`povDelimiter="_"`。
    除了默认分隔符外，仅支持以下分隔符：`_`（下划线）、`#`（井号）、`&`（和号）、`~`（波形符）、`%`（百分比）、`;`（分号）、`:`（冒号）、`-`（短划线）。
  - `clearInput=true|false` 指定是否清除输入数据。默认值为 `false`。
  - `clearAllocatedValues=true|false` 指定是否清除分配的值。默认值为 `false`。
  - `clearAdjustmentValues=true|false` 指定是否清除调整值。默认值为 `false`。

> **Note:**
>
> * 参数值（`true` 或 `false`）必须全部小写。
> * `clearInput`、`clearAllocatedValues` 或 `clearAdjustmentValues` 参数中的至少一个必须设置为 `true`。

示例

- 使用默认 POV 分隔符从默认 PCM_CLC 多维数据集中的 POV 清除数据：
  ```
  epmAutomate clearDataByPointOfView FY21::Jan::Actual::Working clearInput=true
  clearAllocatedValues=true clearAdjustmentValues=true
  ```
- 使用自定义 POV 分隔符从特定多维数据集中的 POV 清除输入数据和分配的值：
  ```
  epmAutomate clearDataByPointOfView FY21_Jan_Actual_Working cubeName=PCM_REP
  povDelimiter="_" clearInput=true clearAllocatedValues=true
  ```
- 使用自定义 POV 分隔符从特定多维数据集中的 POV 清除输入数据：

```
epmAutomate clearDataByPointOfView FY21:Jan:Actual:Working cubeName=PCM_REP
povDelimiter=":" clearInput=true
```

clearDataByProfile

从 Financial Consolidation and Close 和 Tax Reporting 中定义的在“清除数据配置文件”中标识的项（例如区域）清除数据。

适用于

Financial Consolidation and Close、Tax Reporting

所需角色

服务管理员

用法

`epmautomate clearDataByProfile PROFILE_NAME`，其中 *PROFILE_NAME* 是“清除数据配置文件”的名称。

示例

```
epmautomate clearDataByProfile clearDataProfile_01
```

clearPOV

从视点 (POV) 组合清除模型对象和数据，或者清除 Profitability and Cost Management 应用程序中 POV 内的数据区域。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate clearPOV APPLICATION_NAME POV_NAME [QUERY_NAME] PARAMETER=VALUE stringDelimiter="DELIMITER"`，其中：

- *APPLICATION_NAME* 是 Profitability and Cost Management 应用程序的名称
- *POV_NAME* 是应用程序中的视点。此值是必需的。
- QUERY_NAME（可选），是与 Profitability and Cost Management 中定义相同的查询的名称。如果指定该名称，则此查询将用于清除 POV 内的数据区域。

> **注：**
>
> 如果指定查询名称，则必须将所有运行时参数（见下方）的值设置为 false。

- PARAMETER=*VALUE* 指明用于清除视点的运行时参数及其值。指定进程所需数量的参数和值对。有效参数（至少需要其中一个）及其值：

  - `isManageRule=true|false` 指定是否清除规则
  - `isInputData=true|false` 指定是否清除输入数据
  - `isAllocatedValues=true|false` 指定是否清除分配值
  - `isAdjustmentValues=true|false` 指定是否清除调整值

> **注：**
>
> 参数值（`true` 或 `false`）必须全部小写。
>
> 要清除 POV 中的数据区域（如果指定了 `QUERY_NAME`），您必须将运行时参数（`isManageRule`、`isInputData`、`isAllocatedValues` 和 `isAdjustmentValues`）的值设置为 `false`。

- `stringDelimiter="DELIMITER"` 指定 POV 值中使用的分隔符。分隔符必须用双引号括起来。默认值为 _（下划线）

示例

- 从 POV 清除所有模型对象和数据：`epmautomate clearPOV BksML12 2012_Jan_Actual isManageRule=true isInputData=true isAllocatedValues=true isAdjustmentValues=true stringDelimiter="_"`
- 清除 POV 内的数据区域：`epmautomate clearPOV BksML12 2012_Jan_Actual queryName=BksML12_2012_Jan_clear_query isManageRule=false isInputData=false isAllocatedValues=false isAdjustmentValues=false stringDelimiter="_"`

cloneEnvironment

克隆当前环境和（可选）身份域对象（用户和预定义角色分配）、数据管理记录、审核记录、作业控制台记录、应用程序属性、收件箱和发件箱的内容以及存储的快照。此命令可以替代迁移中的克隆环境功能。

在源或目标环境的预定日常维护时间之后启动克隆操作。如果在克隆正在进行中时开始对源环境进行日常维护，则克隆过程将终止。即使在日常维护开始时克隆正在进行，目标环境上的克隆过程也不会受到影响。在这种情况下，将在克隆完成后运行日常维护。

如果克隆环境需要很长时间，请重新调度源环境上的日常维护开始时间，以避免克隆过程被终止。有关重置日常维护开始时间的信息，请参阅以下信息源。

- setDailyMaintenanceStartTime
- 《管理员入门指南》中的“管理日常维护”。
- 《*REST APIs*》指南中的 "Viewing and Setting the Daily Maintenance Window Time"。

Note:

- **数据管理：** 如果临时表包含大量记录，克隆数据管理记录可能需要很长时间。类似地，克隆收件箱和发件箱的内容以及存储的快照可能需要大量时间，尤其当它们包含大量数据时更是如此。
- **旧环境：**克隆将保持当前 Oracle Essbase 版本，如以下场景中所述：
  - 场景 1：您克隆的源旧环境使用的 Essbase 版本不支持混合多维数据集，目标旧环境使用的 Essbase 版本支持混合多维数据集。在此场景中，目标环境中的 Essbase 将降级，以匹配源环境中的版本。
  - 场景 2：您克隆的源旧环境使用的 Essbase 版本支持混合多维数据集，目标旧环境使用的 Essbase 版本不支持混合多维数据集。在此场景中，目标环境中的 Essbase 将升级，以匹配源环境中的版本。
  - 场景 3：您克隆的源旧环境使用的 Essbase 版本不支持混合多维数据集，目标 EPM Standard Cloud Service 或 EPM Enterprise Cloud Service 环境默认使用的 Essbase 版本支持混合多维数据集。在此场景中，目标环境中的 Essbase 不会降级，即不会匹配源环境中的版本。
- **Planning：**如果 Planning 业务流程包含重命名的植入期间成员而原始成员已被自定义期间成员取代，克隆可能会失败。例如，您将植入的 *YearTotal* 期间成员重命名为 *unused_YearTotal* 后，添加了使用原始植入成员名称（在此示例中为 *YearTotal*）的备用类型期间成员。在这种情况下，环境克隆可能会失败。

有关这些主题的详细信息，请参阅《管理迁移》中的“克隆云 EPM 环境”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

身份域管理员角色是克隆用户和预定义角色所必需的。

用法

`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] [ApplicationProperties=true|false]`，其中：

Note:

- `dataManagement` 参数不适用于 Oracle Enterprise Data Management Cloud 和 Narrative Reporting 环境。
  仅当源环境和目标环境的每月更新相同或目标环境比源环境多更新一次时，才可以克隆数据管理记录。例如，22.01 数据管理记录可以克隆到其他 22.01 环境，或者仅克隆到 22.02 环境。
- `jobConsole` 参数仅适用于 Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。
- `appAudit` 参数仅适用于 Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。
  默认情况下，Financial Consolidation and Close 和 Tax Reporting 的审核信息包含在快照中。
- 如果 `dataManagement`、`jobConsole` 或 `appAudit` 参数不适用于环境，该命令将忽略您指定的值。
- `ApplicationProperties` 参数仅适用于 Account Reconciliation。

- *TARGET_USERNAME* 是目标环境中的服务管理员的 ID。您必须使用目标身份域用户名（而非 SSO 用户名）。如果您计划在目标环境中克隆用户和角色分配，则此用户还必须具有身份域管理员角色。
- *TARGET_PASSWORD* 是 *TARGET_USERNAME* 标识的用户的加密密码文件的位置。
- *TARGET_URL* 是要成为克隆环境的环境 `CLOUD-EPM_BASE_URL`。
- *SnapshotName*（可选）是应该用于克隆的快照的名称。此快照必须存在于源环境中。默认值为 `Artifact Snapshot`，即使用上次的维护快照来克隆环境。
- *UsersAndPreDefinedRoles*（可选）标识是否克隆用户及其预定义角色分配（始终克隆访问控制组）。默认值为 `false`。
  要让此选项生效，*TARGET_USER_NAME* 标识的用户必须在目标环境中具有身份域管理员角色。

  如果在选中此复选框后，克隆环境的用户不是身份域管理员，则导入用户及其预定义角色将失败。迁移状态报表中将记录以下错误：无法导入外部目录对象 `<artifact_name>`。用户 `<user_name>` 无权执行此操作。用户必须具有身份域管理员角色才能执行此操作。

  – 如果您不导入用户，并且源快照中的用户未分配目标环境中的预定义角色，则将显示错误 (`EPMIE-00070: Failed to find user during assigned roles import`)。
  – 不会克隆身份域管理员角色分配。仅分配了身份域管理员角色的用户不会克隆到目标环境。
    将克隆在源环境中分配了身份域管理员角色和预定义角色的用户，但在目标环境中仅为这些用户分配各自的预定义角色。这些用户在目标环境中将没有身份域管理员角色。
  – 对用户预定义角色的更改将根据源快照中分配的角色进行更新。但是，不会为了匹配源快照中的角色分配而删除目标中的角色分配。例如，假设 `jdoe` 分配给目标环境中的超级用户预定义角色，但在源快照中只有用户角色。在这种情况下，此命令会将 `jdoe` 分配给用户角色，并且不会删除目标环境中的超级用户角色分配。

  - 如果源快照中不存在现有用户，此命令不会从目标环境中删除这些用户。例如，jdoe 在目标环境中有一个帐户，但此帐户在源快照中不存在。在这种情况下，目标环境中 jdoe 的该帐户不会被删除。
  - 此命令添加目标环境中不存在的用户；它不会更新目标环境中的当前用户属性，即使源快照中的这些属性不同。例如，如果源快照中 jdoe 的姓氏在目标环境中拼写不同，则不会在目标环境中进行更改。随机密码将分配给目标环境中的新用户。新用户将收到提示其更改密码的帐户激活电子邮件。
  - 此命令不会更改目标环境中现有用户的密码，即使它在源快照中有所不同。

- dataManagement=true|false（可选）将源环境中的数据管理记录克隆到目标环境。默认值为 true，即克隆数据管理记录。如果不希望克隆数据管理记录，请将此值设置为 false。
- appAudit=true|false（可选）将源环境中的审核记录克隆到目标环境。默认值为 true，即克隆应用程序审核数据。如果不希望将应用程序审核数据克隆到目标环境，请将此值设置为 false。
- jobConsole=true|false（可选）将源环境中的作业控制台记录克隆到目标环境。默认值为 true。如果不希望克隆作业控制台记录，请将此值设置为 false。
- storedSnapshotsAndFiles（可选）标识命令是否应该克隆收件箱和发件箱的内容以及存储的快照。默认值为 false。

> **Note:**
>
> 仅克隆收件箱和发件箱中的顶层文件夹，而不克隆子文件夹。如果需要保留子文件夹的内容，请将它们备份到本地计算机，然后将它们上传到目标环境。

- *DailyMaintenanceStartTime*（可选）将克隆的目标环境的维护开始时间重置为源环境的维护开始时间。默认值为 true。要保持目标环境的当前维护开始时间，请将此值设置为 false。
- ApplicationProperties（可选）克隆 Account Reconciliation 应用程序设置（Redwood 体验、主题、业务流程名称、标识图像和背景图像）。默认值为 true。
  为了使目标环境保留其当前的应用程序设置，请将此值设置为 false。

示例

- 克隆环境、用户和预定义角色分配、审核数据、作业控制台记录以及数据管理记录。此外，还将目标环境的维护开始时间更改为源环境的维护开始时间：
  ```
  epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com UsersAndPreDefinedRoles=true
  ```
- 克隆环境（包括收件箱和发件箱的内容以及存储的快照），但不克隆用户和预定义角色分配、数据管理记录、审核记录以及作业控制台记录，并且不更改目标环境的维护开始时间：
  ```
  epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com DataManagement=false appAudit=false jobConsole=false storedSnapshotsAndFiles=true DailyMaintenanceStartTime=false
  ```
- 使用自定义快照克隆整个环境（用户和预定义角色分配、审核数据、作业控制台记录、收件箱和发件箱内容、存储的快照以及数据管理记录）。此外，还将目标环境的维护开始时间更改为源环境的维护开始时间：
  ```
  epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-cloudpln.pbcs.us1.oraclecloud.com UsersAndPreDefinedRoles=true storedSnapshotsAndFiles=true SnapshotName=SampleSnapshot
  ```
- 仅限 **Account Reconciliation**：克隆环境、用户和预定义角色分配，以及数据管理记录，但保留目标环境的应用程序设置：

```
epmAutomate cloneEnvironment serviceAdmin Password.epw https://test-
cloudarcs.arcs.epm.us.oraclecloud.com UsersAndPreDefinedRoles=true
ApplicationProperties=false
```

compactCube

使用压缩多维数据集类型的作业中定义的设置压缩聚合存储 (ASO) 多维数据集的大纲文件。压缩大纲有助于使多维数据集大纲文件保持最佳大小。压缩不会清除多维数据集中的数据或将业务流程中的更改推送到 Oracle Essbase。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、销售规划和战略性人员规划

所需角色

服务管理员

用法

`epmautomate compactCube` *`ASO_CUBE_NAME`*，其中 *`ASO_CUBE_NAME`* 是要压缩其大纲的 ASO 多维数据集的名称。

示例

```
epmautomate compactCube Vis1Aso
```

copyDataByPointOfView

将数据从多维数据集中的源 POV 复制到同一或另一个 Enterprise Profitability and Cost Management 多维数据集中的目标 POV。

适用于

Enterprise Profitability and Cost Management

所需角色

服务管理员

用法

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

- *`SOURCE_POV_NAME`* 是要从中复制数据的源 POV 的名称。
- *`TARGET_POV_NAME`* 是源中的数据要复制到的有效目标 POV 的名称。
- `copyType` 标识要从源 POV 复制的数据。有效值为：
  - `ALL_DATA`，将所有输入和计算的数据复制到目标 POV。
  - `INPUT`，将所有输入数据（包括动因数据）复制到目标 POV。
- *`SOURCE_CUBE_NAME`* 是包含源 POV 的多维数据集的名称。
- *`TARGET_CUBE_NAME`* 是包含目标 POV 的多维数据集的名称。

- `PARAMETER=VALUE` 表示可选的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：
  - `povDelimiter`（可选）是 POV 值中使用的分隔符。默认值为 `:`（双冒号）。此值必须用双引号括起来。示例：`povDelimiter="_"`。
    除了默认分隔符外，仅支持以下分隔符：`_`（下划线）、`#`（井号）、`&`（和号）、`~`（波形符）、`%`（百分比）、`;`（分号）、`:`（冒号）、`-`（短划线）。
  - `createDestPOV=true|false` 指定目标 POV 不存在时是否创建该 POV。默认值为 `false`。如果目标 POV 不存在，则必须将此参数值设置为 `true`。

示例

- 将所有数据复制到同一多维数据集中的其他 POV：
```
epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
FY22_Jan_Actual_Working ALL_DATA PCM_CLC PCM_CLC povDelimiter="_"
createDestPOV=true
```
- 将所有数据复制到其他多维数据集中的其他 POV：
```
epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
FY22_Jan_Actual_Working ALL_DATA PCM_CLC PCM_REP povDelimiter="_"
createDestPOV=true
```
- 将输入数据复制到同一多维数据集中的其他 POV：
```
epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
FY22_Jan_Actual_Working INPUT PCM_CLC PCM_CLC povDelimiter="_"
createDestPOV=true
```
- 将输入数据复制到其他多维数据集中的其他 POV：
```
epmAutomate copyDataByPointOfView FY21_Jan_Actual_Working
FY22_Jan_Actual_Working INPUT PCM_CLC PCM_REP povDelimiter="_"
createDestPOV=true
```

copyDataByProfile

复制“复制数据配置文件”中标识的项（例如区域）的数据。

适用于

Financial Consolidation and Close、Tax Reporting

所需角色

服务管理员

用法

`epmautomate copyDataByProfile PROFILE_NAME`，其中 `PROFILE_NAME` 是 Financial Consolidation and Close 和 Tax Reporting 中定义的“复制数据配置文件”的名称。

示例

```
epmautomate copyDataByProfile copyDataProfile_01
```

copyFileFromInstance

将文件从源环境复制到从中执行此命令的环境。

运行此命令之前，使用 EPM Automate 登录到要将文件复制到的环境。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate copyFileFromInstance` *`SOURCE_FILE_NAME USERNAME PASSWORD_FILE URL TARGET_FILE_NAME`*，其中：

- *`SOURCE_FILE__NAME`* 是要从源环境复制的文件的名称（包括扩展名）。
- *`USERNAME`* 是源环境的服务管理员的用户名。
- *`PASSWORD_FILE`* 是包含源环境中服务管理员加密密码的文件的名称和位置。
- *`URL`* 是源环境的 URL。
- *`TARGET_FILE_NAME`* 是从中运行此命令的环境中的文件的唯一名称（包括扩展名）。

示例

```
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

将文件或备份快照从 Oracle Object Storage 存储桶复制到当前环境。

如果您正在复制备份快照，则此命令从对象存储桶复制该快照并在 Oracle Fusion Cloud Enterprise Performance Management 中提取其内容。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate copyFromObjectStorage` *`USERNAME PASSWORD URL TARGET_FILE_NAME`*，其中：

- *`USERNAME`* 是在 Oracle Object Storage Cloud 中具有所需访问权限的用户的 ID。

对于在联合身份提供程序中创建的用户，请指定用户的全限定名称（例如 `exampleIdP/jdoe` 或 `exampleIdP/john.doe@example.com`，其中 `exampleIdP` 是联合身份提供程序的名称）。对于其他用户，请提供用户 ID。

- *PASSWORD* 是与用户关联的 Swift 密码或身份验证令牌。此密码与您用于登录到对象存储控制台的密码不同。身份验证令牌是 Oracle 生成的令牌，您可以将其与第三方 API 一起用于身份验证，例如与 Swift 客户端一起用于身份验证。有关创建此令牌的说明，请参阅 *Oracle Cloud Infrastructure* 文档 中的“创建身份验证令牌”。
- *URL* 是 Oracle Object Storage Cloud 存储桶的 URL，包括存储桶名称和要复制的对象名称。
  URL 格式：

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

  此 URL 的组成部分：

  - `region_identifier` 是 Oracle Cloud Infrastructure 托管区域。
  - `namespace` 是所有存储桶和对象的顶级容器。在创建帐户时，会为每个 Oracle Cloud Infrastructure 租户分配一个唯一的对象存储名称空间名称，该名称由系统生成而且不可变。租户的名称空间名称（例如，`axaxnpcrorw5`）在所有区域中有效。
  - `bucket_name` 是用于存储数据和文件的逻辑容器的名称。存储桶按区间进行组织和维护。系统生成的存储桶名称（例如， `bucket-20210301-1359`）反映当前的年份、月份、日期和时间。
  - `object_name` 是要从 Oracle Object Storage Cloud 复制的快照或文件的名称。此值必须与 Object Storage Cloud 中的对象完整名称完全匹配。不使用 `.zip` 等扩展名，除非对象名称包含它。

  有关详细信息，请参阅 *Oracle Cloud Infrastructure* 文档中的以下主题

  - 区域和可用性域
  - 了解对象存储名称空间
  - 管理存储桶
- *TARGET_FILE_NAME* 是云 EPM 环境中文件或快照的唯一名称。在复制快照时，请勿指定 ZIP 扩展名，以便可以将此文件名用于 importSnapshot 命令。
  超过 100 MB 的文件存储在 Oracle Object Storage 的逻辑目录中，该目录同时存储用于标识文件段的清单文件。按照 *TARGET_FILE_NAME* 指定逻辑目录的名称。

示例

在这些示例中，将 *URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET* 替换为采用如下格式的有效 URL：`https://swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/bucket_name/`。

- 将名为 `backup_Snapshot_12_05_20.zip` 的快照从 Oracle Object Storage 存储桶复制到云 EPM 并对其进行重命名：

```
epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd
URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/backup_Snapshot_12_05_20.zip
snapshot_from_osc
```

- 将名为 `backup_Snapshot_12_05_20` 的快照从 Oracle Object Storage 存储桶复制到云 EPM 并对其进行重命名：

```
epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd
URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/backup_Snapshot_12_05_20
snapshot_from_osc
```

- 将名为 backup_Snapshot_12_05_20 的快照从 Oracle Object Storage 存储桶复制到云 EPM，不对其进行重命名：
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd *URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET*/backup_snapshot_12_05_20 backup_snapshot_12_05_20
- 将文件从 Oracle Object Storage 存储桶复制到云 EPM：
  epmautomate copyFromObjectStorage oracleidentitycloudservice/jDoe example_pwd *URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET*/example_file.txt copied_from_osc.txt

copyFromSFTP

将文件直接从 SFTP 服务器复制到 OCI（第 2 代）环境。此命令可用于将数据文件从安全 SFTP 服务器直接复制到 Oracle Fusion Cloud Enterprise Performance Management 环境。使用此命令，您可以直接从 SFTP 服务器复制文件，而无需先使用 SFTP 客户端将其下载到另一个服务器，然后使用 EPM Automate 或 REST API 将其上传到云 EPM。

如果为 SFTP 服务器启用了 IP 允许列表，则必须将托管云 EPM 环境的 OCI 区域的出站 IP 地址添加到 SFTP 服务器的 IP 允许列表。有关 OCI 区域的出站 IP 地址，请参阅《运维指南》中的“云 EPM 数据中心和区域的出站 IP 地址”。

> **Note:**
>
> 此命令仅支持在端口 22 上运行的 SFTP 服务器。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

服务管理员

用法

epmautomate copyFromSftp *SFTP_SERVER_URL FILE_NAME* username=*USERNAME* password=*PASSWORD*，其中：

- *SFTP_SERVER_URL* 是 SFTP 服务器的 URL，其中包括要从服务器复制的文件的位置和名称。
- *FILE_NAME* 是云 EPM 环境中文件的文件名。例如，如果您不复制到默认上传位置，则指定路径。
- *USERNAME* 是对 SFTP 服务器具有所需访问权限的用户的 ID。
- *PASSWORD* 是 SFTP 用户密码。

示例

```
epmautomate copyFromSftp sftp://mySFTP_Server/someDirectory/someFile.csv
myFile.csv username=jdoe password=Sftpwelcome
```

copyOwnershipDataToNextYear

将所有权数据从一年中的最后一个期间复制到下一年的第一个期间。

初始默认设置和覆盖所有权设置会自动从同一年份中的先前期间结转至后续期间，但不会结转至后续年份中的期间。要将一年中最后一个期间的最新所有权设置结转至下一年的第一个期间，必须将 POV 中该年份的最后一个期间的所有权设置复制到下一年的第一个期间。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 用户

用法

`epmautomate copyOwnershipDataToNextYear` *`Scenario Year`*，其中：

- `Scenario` 是要从中复制所有权数据的方案的名称。
- `Year` 是要从中将所有权数据复制到下一年第一个期间的年份。

示例

```
epmautomate copyOwnershipDataToNextYear FCCS_total_Actual FY18
```

copyPOV

将模型对象和 Oracle Essbase 多维数据集从源 POV 复制到目标 POV。

适用于

Profitability and Cost Management。

所需角色

- 服务管理员
- 超级用户

用法

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

- *`APPLICATION_NAME`* 是包含源 POV 的 Profitability and Cost Management 应用程序的名称。
- *`SOURCE_POV_NAME`* 是指定应用程序中的源视点的名称
- *`TARGET_POV_NAME`* 是 `Draft` 状态的有效目标视点的名称
- `PARAMETER=`*`VALUE`* 指明用于复制视点的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：

  - `isManageRule=true|false` 指定是否复制规则。
  - `isInputData=true | isAllData=true | isAllInputData=true` 可以指定如何复制数据。对于这些参数，默认值为 false。仅将其中一项指定为 true：
    * 指定 `isInputData=true` 将输入数据复制到目标 POV。
    * 指定 `isAllData=true` 将所有输入和计算数据复制到目标 POV。
    * `AllInputData=true` 将所有输入数据（包括动因数据）复制到目标 POV。
  - `modelViewName=`*`NAME`* 指定要从源 POV 复制到目标 POV 的数据切片的名称。
  - `createDestPOV=true|false` 指定目标 POV 不存在时是否创建该 POV。
  - `nonEmptyTupleEnabled=true|false` 指定是否启用非空元组 (Non-Empty Tuple, NET)，以便此命令仅考虑具有数据的交叉点。默认值为 `true`，在极少数情况下，该默认值可能会导致在复制 Essbase 数据时性能不佳。在这些情况下，可以使用 `nonEmptyTupleEnabled=false` 来覆盖默认值，从而改进性能。

> **注：**
>
> 参数值（`true` 或 `false`）必须全部小写。

- `stringDelimiter="`*`DELIMITER`*`"` 指定 POV 值中使用的分隔符。分隔符必须用双引号括起来。

示例

- `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

将当前快照从源环境复制到从中运行此命令的目标环境。

此命令主要用作迁移环境的第一步，即将当前快照从一个环境（例如测试环境）复制到生产环境。使用 importSnapshot 命令完成迁移过程。

运行此命令之前，启动 EPM Automate 会话并登录到目标环境。

如果在生成源环境的快照时执行此命令来复制当前快照（例如，在日常维护期间），您会收到 `File not found`（找不到文件）错误。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate copySnapshotFromInstance SNAPSHOT_NAME USERNAME PASSWORD_FILE URL`，其中：

- *SNAPSHOT_NAME* 是源环境中现有快照的名称。
- *USERNAME* 是源环境的服务管理员的用户名。
- *PASSWORD_FILE* 是包含源环境中服务管理员加密密码的文件的名称和位置。
- *URL* 是源环境的 URL。

示例

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

copyToObjectStorage

将文件或快照从当前环境复制到 Oracle Object Storage Cloud 存储桶。

如果要复制快照，则此命令会先压缩快照内容，然后再将其复制到 Oracle Object Storage。

为了支持快速复制文件，此命令将大文件（超过 100 MB）分成多个 10 MB 的段（命名为 *FILE_NAME*/*FILE_NAME*_object_store_bytes_seg_0 至 *FILE_NAME*/*FILE_NAME*_object_store_bytes_seg_*n*）并创建清单文件（命名为 *FILE_NAME*/*FILE_NAME*.manifest）。文件段与清单文件一起存储在 Oracle Object Storage 中。在对象存储控制台中，文件显示为包含文件段和清单文件的逻辑目录。

小于 100 MB 的文件不分段，并以原始文件名存储。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate copyToObjectStorage SOURCE_FILE_NAME USERNAME PASSWORD URL`，其中：

- *SOURCE_FILE_NAME* 是 Oracle Fusion Cloud Enterprise Performance Management 中文件或快照的名称。如果要复制快照，则不要指定 ZIP 扩展名。
- *USERNAME* 是对 Oracle Object Storage Cloud 具有所需写入访问权限的用户的 ID。

对于在联合身份提供程序中创建的用户，请指定用户的全限定名称（例如 `exampleIdP/jdoe` 或 `exampleIdP/john.doe@example.com`，其中 `exampleIdP` 是联合身份提供程序的名称）。对于其他用户，请提供用户 ID。

- *PASSWORD* 是与用户关联的 Swift 密码或身份验证令牌。此密码与您用于登录到对象存储控制台的密码不同。身份验证令牌是 Oracle 生成的令牌，您可以将其与第三方 API 一起用于身份验证，例如与 Swift 客户端一起用于身份验证。有关创建此令牌的说明，请参阅 *Oracle Cloud Infrastructure* 文档 中的“创建身份验证令牌”。
- *URL* 是 Oracle Object Storage Cloud 存储桶 URL，后面可以附加对象名称。
  不包含对象名称的 URL 格式：

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

  包含对象名称的 URL 格式：

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

  此 URL 的组成部分：

  - `region_identifier` 是 Oracle Cloud Infrastructure 托管区域。
  - `namespace` 是所有存储桶和对象的顶级容器。在创建帐户时，会为每个 Oracle Cloud Infrastructure 租户分配一个唯一的对象存储名称空间名称，该名称由系统生成而且不可变。租户的名称空间名称（例如，`axaxnpcrorw5`）在所有区域中有效。
  - `bucket_name` 是用于存储数据和文件的逻辑容器的名称。存储桶按区间进行组织和维护。系统生成的存储桶名称（例如， `bucket-20210301-1359`）反映当前的年份、月份、日期和时间。
  - `object_name`（可选）是要用于 Oracle Object Storage Cloud 上文件的名称。如果未指定对象名称，则将使用文件的原始名称复制该文件。

  有关详细信息，请参阅 *Oracle Cloud Infrastructure* 文档中的以下主题

  - 区域和可用性域
  - 了解对象存储名称空间
  - 管理存储桶

示例

在这些示例中，将 *URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET* 替换为采用如下格式的有效 URL：`https://swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/bucket_name/`。

- 将快照复制到对象存储的存储桶并对其进行重命名：
  ```
  epmautomate copyToObjectStorage "Artifact Snapshot"
  oracleidentitycloudservice/jDoe example_pwd
  URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/Snapshot_04_30_21
  ```
- 将文件复制到对象存储的存储桶：
  ```
  epmautomate copyToObjectStorage example_file.txt oracleidentitycloudservice/
  jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET
  ```
- 将文件复制到对象存储的存储桶并对其进行重命名：
  ```
  epmautomate copyToObjectStorage example_file.txt eoracleidentitycloudservice/
  jDoe example_pwd URL_OF_THE_ORACLE_OBJECT_STORAGE_BUCKET/epm_text_file.txt
  ```

copyToSFTP

将文件从 OCI（第 2 代）环境中的位置复制到安全 FTP 服务器上的位置。使用此命令，您可以将文件直接从 Oracle Fusion Cloud Enterprise Performance Management 环境复制到 SFTP 服务器，而无需先使用 EPM Automate 或 REST API 将其下载到另一个服务器，然后使用 SFTP 客户端将其上传到 SFTP 服务器。

如果为 SFTP 服务器启用了 IP 允许列表，则必须将托管云 EPM 环境的 OCI 区域的出站 IP 地址添加到 SFTP 服务器的 IP 允许列表。有关 OCI 区域的出站 IP 地址，请参阅《运维指南》中的“云 EPM 数据中心和区域的出站 IP 地址”。

> **Note:**
>
> 此命令仅支持在端口 22 上运行的 SFTP 服务器。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate copyToSftp` *`SFTP_SERVER_URL EPM_FILE_NAME`* `username=`*`USERNAME`* `password=`*`PASSWORD`*，其中：

- *`SFTP_SERVER_URL`* 是 SFTP 服务器的 URL，其中包括要复制文件的目录和文件名。
- *`EPM_FILE_NAME`* 是要从云 EPM 环境复制的文件的名称和位置（如果不是来自默认发件箱）。如果要复制快照，则不要指定 .ZIP 扩展名
- *`USERNAME`* 是对 SFTP 服务器具有所需访问权限的用户的 ID。
- *`PASSWORD`* 是 SFTP 用户密码。

示例

```
epmautomate copyToSftp sftp://mySFTP_Server/someDirectory/some_file.csv
myFile.csv username=jDoe password=SftpWelcome
```

createGroups

使用上传到环境中的 ANSI 或 UTF-8 编码的 CSV 文件将组添加到访问控制。

服务管理员使用 uploadFile 命令将文件上传到环境中。文件格式如下：

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

组名不区分大小写。完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate createGroups FILE_NAME`，其中 *`FILE_NAME`* 是 CSV 文件的名称，该文件包含组名称和说明。

示例

```
epmautomate createGroups group_file.CSV
```

createNRSnapshot

为 Narrative Reporting 环境创建名为 `EPRCS_Backup.tar.gz` 的按需快照。

您可以使用 downloadFile 命令将 `EPRCS_Backup.tar.gz` 和错误文件下载到本地计算机，或者使用 copyFileFromInstance 命令将其复制到其他环境。

`EPRCS_Backup.tar.gz` 中的应用程序数据截至上次日常维护日期。如果需要备份最新数据，请使用 Narrative Reporting 数据导出功能。

适用于

Narrative Reporting

所需角色

服务管理员

用法

`epmautomate createNRSnapshot [errorFile=Error_File.txt]`，其中 `errorFile`（可选）标识用于记录错误的唯一文本文件的名称（如果命令遇到任何错误）。

示例

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

createReconciliations

将配置文件复制到指定的期间。

适用于

Account Reconciliation。

所需角色

- 服务管理员
- 任何预定义角色（查看者角色需要“Reconciliation - 编制者”应用程序角色）

用法

`epmautomate createreconciliations` *`PERIOD SAVED_FILTER`*，其中：

- *`PERIOD`* 是期间名称
- *`SAVED_FILTER`* 是保存的公共筛选器的名称。如果未指定保存的筛选器，则该命令会复制所有适用的配置文件

示例

- 复制期间的所有配置文件：`epmautomate createReconciliations "January 2015"`
- 复制特定筛选器的配置文件：`epmautomate createReconciliations "January 2015" "Corporate Recs"`

deleteFile

从默认上传位置、收件箱或发件箱、数据管理文件夹或 profitinbox/profitoutbox 删除文件或快照。

要从默认上传位置之外的位置删除文件，必须指定文件位置。

如果执行此命令以删除正在生成或存档的快照，您将收到以下错误之一：

- 如果正在生成快照：`File not found`（找不到文件）
- 如果正在对快照进行存档：`Archive process is in progress. Unable to Rename or Delete`（正在存档。无法重命名或删除）

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate deleteFile` *`FILE_NAME`*

> **注：**
>
> 必须指定包含扩展名的文件名；例如 `data.csv`、`data.zip`（如果适用）。您可以删除快照，而不用指定其文件扩展名 (.ZIP)。但是，这种用法已过时。如果文件不在默认位置，则应指定文件的位置。有关详细信息，请参阅“默认文件位置”。支持的位置包括 `inbox`、`profitinbox`、`outbox`、`profitoutbox`、`to_be_imported` 和 `inbox/directory_name`。

示例

- 从默认上传位置删除文件：
  `epmautomate deleteFile data.csv`
- 从收件箱删除文件：
  `epmautomate deleteFile inbox/data.csv`
- 从发件箱删除：
  `epmautomate deleteFile outbox/data.csv`
- 删除使用迁移创建的快照：
  - `epmautomate deleteFile "Backup 18-06-12.zip"` 或
  - `epmautomate deleteFile "Backup 18-06-12"`（已过时）
- 从 profitinbox (Profitability and Cost Management) 删除：
  `epmautomate deleteFile profitinbox/data.csv`
- 从 profitoutbox (Profitability and Cost Management) 删除：
  `epmautomate deleteFile profitoutbox/data.csv`
- 从数据管理上传文件夹删除：
  `epmautomate deleteFile inbox/dm_data/data.csv`
- 从数据管理文件夹删除：
  `epmautomate deleteFile outbox/dm_data/data.csv`

deleteGroups

基于上传到环境中的 ANSI 或 UTF-8 编码的 CSV 文件中提供的信息，从访问控制中删除组。

服务管理员使用 uploadFile 命令将文件上传到环境中。文件格式如下：

```
Group Name
Example_grp1
Example_grp2
```

文件中的组名值不区分大小写。完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate deleteGroups` *`FILE_NAME`*，其中 *`FILE_NAME`* 是 CSV 文件的名称，该文件包含要从访问控制中删除的组的名称。

示例

```
epmautomate deleteGroups group_file.CSV
```

deletePointOfView

从 Enterprise Profitability and Cost Management 应用程序中的 POV 中删除对象和 Oracle Essbase 多维数据集数据。

适用于

Enterprise Profitability and Cost Management

所需角色

服务管理员

用法

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

- *`POV_NAME`* 标识要删除的 POV 的名称。
- `povDelimiter` 是 POV 值中使用的分隔符。默认值为 `::`（双冒号）。此值必须用双引号括起来。示例：`povDelimiter="_"`。
  除了默认分隔符外，仅支持以下分隔符：`_`（下划线）、`#`（井号）、`&`（和号）、`~`（波形符）、`%`（百分比）、`;`（分号）、`:`（冒号）、`-`（短划线）。

示例

- 删除使用自定义 POV 分隔符的 POV
  ```
  epmAutomate deletePointOfView FY21_Jan_Actual_Working povDelimiter="_"
  ```
- 删除使用默认 POV 分隔符的 POV
  ```
  epmAutomate deletePointOfView FY21::Jan::Actual::Working
  ```

deletePOV

从 Profitability and Cost Management 中的 POV 中删除模型对象和 Oracle Essbase 多维数据集数据。

适用于

Profitability and Cost Management

所需角色

- 服务管理员

- 超级用户

用法

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

- *APPLICATION_NAME* 是包含要删除 POV 的 Profitability and Cost Management 应用程序的名称。
- *POV_NAME* 是要删除的视点名称。此值是必需的。
- `stringDelimiter="*DELIMITER*"` 指定 POV 值中使用的分隔符。分隔符必须用双引号括起来。

示例

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

deployCube

部署或重新部署 Profitability and Cost Management 应用程序的计算多维数据集。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate deployCube *APPLICATION_NAME* PARAMETER=*VALUE* comment="comment"`，其中：

- *APPLICATION_NAME* 是 Profitability and Cost Management 应用程序的名称
- `PARAMETER=*VALUE*` 指明多维数据集的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：

  > **注：**
  >
  > 参数值（`true` 或 `false`）必须全部小写。

  - `isKeepData=true|false`

    指定是否保留现有数据（如果有）
  - `isReplaceCube=true|false` 指定是否替换现有多维数据集

    > **注：**
    >
    > `isKeepData` 值和 `isReplaceCube` 值不能同时设置为 `true`。

  - `isRunNow=true|false` 指定是否立即运行进程
- `comment` 是用双引号括起来的可选注释

示例

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

deployEJTemplates

部署最终确定的企业日记帐模板以在 Financial Consolidation and Close 中打开期间。部署企业日记帐模板将为所选期间创建与模板关联的经常性日记帐。您还可以使用部署的模板创建即席日记帐。

使用此命令，可取代在月初使用 Financial Consolidation and Close 屏幕部署新的企业日记帐模板。

适用于

Financial Consolidation and Close

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate deployEJTemplates` *`YEAR PERIOD`* `[Template=`*`TEPMPLATE_NAME`*`] [ResetJournals=true|false]`，其中：

- `Year` 是日记帐年份。
- `Period` 是日记帐期间。仅当指定年份时才能指定此值。
- `Template=`*`TEMPLATE_NAME`* 标识要部署的日记帐的名称。要部署多个日记帐，请以 `Template=`*`TEMPLATE_NAME`* 格式提供每个唯一的模板名称，例如 `Template="Loan Details" Template="Housing Details" Template="Repayment Details"`。
  如果未指定此参数值，则该命令将部署指定年份和期间组合的所有模板。
- `ResetJournals`（可选）指示在重新部署模板后是否必须将所有日记帐重置到第一个阶段。默认值为 `false`。
  Financial Consolidation and Close 根据模板更改在内部验证此值，并且可以根据需要覆盖您指定的值。

示例

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

deployFormTemplates

将最终确定的表单模板部署到新的数据收集期间，以创建补充数据表单，从而确保数据收集过程一致且可重复。

适用于

Financial Consolidation and Close、Tax Reporting。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate deployFormTemplates` *`COLLECTION_INTERVAL`* `[`*`DIMENSION`*`] [Template] [resetWorkFlows=true|false]`，其中：

- `COLLECTION_INTERVAL` 是模板要部署到的收集间隔的名称。
- `DIMENSION`（可选）以 *`DIMENSION=MEMBER_NAME`* 格式指定数据收集过程的频率维。指定收集间隔中定义的任意多个维（最大值为四，包括年和期间；例如，`"Year=2020" "Period=July" "Product=Oracle EPM" "Consolidation=entity Input"`）。如果未指定此参数值，则不使用默认值。
- `Template`（可选）以 `Template=`*`TEMPLATE_NAME`* 格式标识要部署的表单模板的唯一名称。可以按此格式指定任意数量的唯一名称（根据需要指定任意多个）。例如，`Template="Loan Details Template" Template="Housing Details Template" Template="Repayment Detals Template"`。
  如果未指定此属性值，则该命令将部署指定间隔的所有模板。
- `resetWorkFlows`（可选）指明所有表单在重新部署后是否重置为第一个阶段。默认值为 `false`。

示例

```
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

将任务管理器模板中的任务部署到任务调度中，以确保一致地执行重复性业务流程。

适用于

Enterprise Profitability and Cost Management、Financial Consolidation and Close 和 Tax Reporting

所需角色

服务管理员

用法

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

- *`TEMPLATE_NAME`* 是要部署的任务管理器模板的名称。
- *`SCHEDULE_NAME`* 是要从模板创建的调度名称。
- *`YEAR`* 是要部署模板的年份维成员。
- *`PERIOD`* 是要部署模板的期间维成员。
- *`DAY_ZERO_DATE`* 是要用于创建调度的基准日期（采用有效格式）。

- `dateFormat`（可选）是基准日期的日期格式。默认格式为 YYYY-MM-DD。
- `orgUnit`（可选）是组织单位的名称。如果未指定值，将使用标准日期映射创建调度。不会使用假日规则。

示例

- 使用基准日期的默认日期格式 (YYYY-MM-DD) 为 *Ind* 组织单位部署任务管理器模板：
  ```
  epmautomate deployTaskManagerTemplate "Vision Monthly Close" "Qtr 2 Close"
  2021 July 2021-07-10 orgUnit=Ind
  ```
- 使用 dd/mm/yyyy 作为基准日期的日期格式，为 Ind 组织单位部署任务管理器模板：
  ```
  epmautomate deployTaskManagerTemplate "Vision Monthly Close" "Qtr 2 Close"
  2021 July 02/07/2021 dateFormat=dd/MM/yyyy orgUnit=Ind
  ```

dismissIPMInsights

运行新的 IPM 洞察作业之前，自动取消智能绩效管理 (IPM) 洞察数据。取消数据会关闭您不打算对其执行操作的所有未完成洞察。此命令是使用 IPM 洞察仪表板手动取消数据的替代方法。

适用于

Planning、Planning 模块、战略性人员规划、销售规划。

所需角色

服务管理员

用法

`epmautomate dismissIPMInsights [comment="`*comment*`"]`，其中 comment（可选）是取消未完成洞察的理由。

示例

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

downloadFile

将环境中的文件下载到本地计算机。

使用此命令下载数据、元数据和备份快照进行本地存储。文件将下载到运行 EPM Automate 的文件夹中。

如果在生成环境快照时执行此命令来下载当前快照（例如，在日常维护期间），您会收到 `File not found`（找不到文件）错误。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员

- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

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

示例

- 下载维护快照：`epmautomate downloadFile "Artifact Snapshot"`
- 下载自定义快照：`epmautomate downloadFile "mySnapshot.zip"`
- 下载 Narrative Reporting 维护快照：`epmautomate downloadFile "EPRCS_Backup.tar.gz"`
- 从默认下载位置下载文件：`epmautomate downloadFile data.csv`
- 从数据管理文件夹下载：`epmautomate downloadfile outbox/dm_data/data.csv`
- 从 profitoutbox 下载：`epmautomate downloadFile profitOutbox/data.csv`

enableApp

启用应用程序。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate enableapp` *`APPLICATION_NAME`*，其中 *`APPLICATION_NAME`* 是要启用的 Profitability and Cost Management 应用程序的名称。

示例

```
epmautomate enableApp BksML12
```

enableQueryTracking

对 ASO 多维数据集启用查询跟踪，以开始捕获用户数据检索模式（查询）。

您可以使用捕获的数据检索模式来优化 ASO 多维数据集聚合。此过程使用 executeAggregationProcess 命令启动。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate enableQueryTracking` *`ASO_CUBE_NAME`*，其中 *`ASO_CUBE_NAME`* 是要激活查询跟踪的 ASO 多维数据集的名称。

示例

```
epmautomate enableQueryTracking VISION_ASO
```

encrypt

使用高级加密标准 (`AES/CBC/PKCS5Padding(256)`) 将 Oracle Fusion Cloud Enterprise Performance Management 密码（或用于访问 OCI（第 2 代）环境的 OAuth2.0 刷新令牌和客户端 ID）和（可选）用于登录到 Oracle Fusion Cloud EPM 环境的 Internet 代理服务器密码加密，并存储在密码文件中。

通过对密钥加密，服务管理员可以与编写 EPM Automate 脚本的开发人员共享其加密的密码文件，以便他们可以执行脚本。这样，将不需要共享服务管理员密码或创建专门用于运行脚本的通用的共享云 EPM 帐户。

密码加密是一次性过程。

> **注：**
>
> 有关对包含特殊字符的密码进行加密的信息，请参阅“处理特殊字符”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

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

- *`PASSWORD|REFRESH_TOKEN PASSWORD`* 是要加密的密码或 OAuth 刷新令牌。您不能在 EPM Automate 中使用公司凭据。
- *`KEY`* 是用于密码加密的私钥。
- *`PASSWORD_FILE`* 是用于存储加密密码或刷新令牌的文件的名称和位置。密码文件必须使用 `.epw` 扩展名。
- `ClientID`（可选）是 OAuth 2.0 安装过程中创建的客户端标识符。加密 OAuth 2.0 刷新令牌时必须指定此值。加密密码时不要指定此值。

- `ProxyServerPassword` 是用于在 HTTP 代理服务器中对用户进行身份验证的密码。仅当在代理服务器上对您的网络启用了身份验证时才需要。

示例

- **仅加密云 EPM 密码：** `epmautomate encrypt P@ssword1 myKey C:\mySecuredir\password.epw`
- **加密云 EPM 和 Internet 代理服务器密码：** `epmautomate encrypt E@xample1 myKey C:\mySecuredir\password.epw ProxyServerPassword=Proxy_Pwd1`
- **加密刷新令牌和客户端 ID：** `epmautomate encrypt AAyyilYBAWD4....FVkxefd8kjoJr6HJPA= myEncyprtion42Key C:\mySecuredir\oauthfile1.epw ClientID=6fdf2e72fd343430ABR22394C`

essbaseBlockAnalysisReport

创建 Essbase 块分析报表，帮助您分析 Oracle Essbase 数据，以支持优化应用程序中的块存储选项 (BSO) 多维数据集（通常用于计算）。

Essbase 块分析报表有助于解决由数据模式导致的性能问题，例如 Essbase BSO 多维数据集中的重复数字。该报表提供有关以下三个方面的信息：

1. **仅包含零的块的百分比**，显示仅包含零的块占导出文件中包含的所有块的百分比
2. **按数字单元格百分比列出的前 10 个重复数字单元格值**，此表以导出文件中所有值的百分比形式显示前 10 个重复值。
3. **具有重复值的前 100 个密集成员组合**，此表显示多维数据集中具有重复值的前 100 个密集组合。**单元格值**列按在层次中出现的顺序将每个成员的值显示为不同的列。例如，如果“期间”跨列，则一月、二月等将对应不同的列。其他密集维出现在行中。这应该有助于您识别重复值的位置。

运行此报表之前，使用 exportEssbaseData 命令将数据从要为其创建块分析报表的多维数据集导出到 zip 文件。您可以根据需要导出 0 级或所有数据。运行此命令为该 zip 文件创建块分析报表。将在发件箱中创建报表；您可以使用 downloadFile 命令将其下载到本地计算机，或者使用 sendMail 命令将其通过电子邮件发送。

适用于

Financial Consolidation and Close、Planning、Planning 模块、自由形式、战略性人员规划、销售规划和 Tax Reporting。

所需角色

服务管理员

用法

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

- `EXPORT_DATA_FILE.zip` 是 zip 文件的名称，该文件包含先前使用 exportEssbaseData 命令从 BSO 多维数据集导出的 Essbase 数据。
- `REPORT_FILE` 是 HTML 格式的块分析报表文件的名称。

示例

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

executeAggregationProcess

使用查询跟踪统计信息（可选）启动聚合过程，以提高 ASO 多维数据集的性能。这是优化 ASO 多维数据集的重要步骤。

在运行此命令之前：

- 使用 enableQueryTracking 命令捕获数据检索统计信息，以优化 ASO 聚合。
- 为业务流程留出足够的时间来捕获用户数据检索模式（查询），此信息可用于创建聚合视图。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

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

- `useQueryData` 表示使用记录的查询数据（使用查询跟踪进行收集）来选择最合适的聚合视图集。默认值为 `false`。
- `includeAlternateRollups` 表示在视图选择过程中包括辅助层次（使用默认级别用法）。默认值为 `disable`。
- `growthSizeRatio`（可选）是在聚合服务器所选视图时多维数据集的最大增长比率。当达到指定的最大增长比率时，多维数据集增长将停止。默认设置允许多维数据集在没有增长比率限制的情况下增长。

> **Note:**
>
> 要创建默认聚合视图，请在不指定可选参数的情况下运行此命令。

示例

- 使用 enableQueryTracking 命令来根据捕获的查询数据创建聚合视图：
  ```
  epmautomate executeAggregationProcess VISION_ASO useQueryData=true
  includeAlternateRollups=enable
  ```
- 创建默认的聚合视图：
  ```
  epmautomate executeAggregationProcess Vis1ASO
  ```

executeBurstDefinition

执行分别输出定义，以指定为一个数据源的单个维的多个成员运行报表或工作簿所需的对象、POV 以及其他设置。

适用于

Narrative Reporting

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

必须通过 ACL 为具有超级用户、用户和查看者角色的用户分配额外的安全性

用法

`epmAutomate executeBurstDefinition` *`ARTIFACT_NAME`*，其中 *`ARTIFACT_NAME`* 是分别输出定义路径和名称。

示例

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

executeReportBurstingDefinition

使用分别输出定义，为单个维的多个成员执行单个报表或工作簿的分别输出，并为每个成员发布 PDF 或静态（在 Oracle Smart View for Office 中不可刷新）Excel 输出。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmAutomate executeReportBurstingDefinition` *`BURST_DEFINITION_NAME`* `[jobName=`*`JOB_NAME`*`]`，其中：

- *`BURST_DEFINITION_NAME`* 是分别输出定义的路径和名称。
- *`JOB_NAME`*（可选）是应该用于执行分别输出定义的作业的名称。默认值为 `Execute Bursting Definition`。

示例

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

exportAccessControl

将用户详细信息报表导出到 CSV 或 XLS 文件，该报表包含有关在环境中具有预定义角色的用户的信息，并列出每个用户的属性（例如名称和电子邮件）以及有关其访问权限的信息（例如分配给组、团队和组织）。

示例报表：

	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			

您可以使用 downloadFile 命令下载此报表。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmAutomate exportAccessControl` *`REPORT_NAME`* `[reportFormat=XLS|CSV]`，其中：

- *`REPORT_NAME`* 是将包含报表的导出文件的名称。
- `reportFormat`（可选）是文件格式。有效值为 XLS 和 CSV（默认值）。

示例

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

exportAppAudit

将数据审核记录导出到 ZIP 文件，您可以下载此文件并将其在本地计算机上存档。环境中最多提供 365 天的审核信息。您可以针对天数或某个日期范围生成此报表。

输出 CSV 文件中的第一个字符是字节顺序标记 (Byte Order Mark, BOM) 字符 `\ufeff`，后跟用双引号括住的加密应用程序标识符。CSV 文件头跟在应用程序标识符后面。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate exportAppAudit EXPORT_FILE_NAME [userNames=USER_NAMES] [nDays=Number_of_Days] [startDate=START_DATE endDate=END_DATE] [excludeApplicationId=true|false]`，其中：

- *EXPORT_FILE_NAME* 是将存储所导出审核数据的 ZIP 文件的名称。可以使用 downloadFile 命令从环境中下载文件。
- `userNames`（可选）是用户登录名的逗号分隔列表。如果指定，则仅导出这些用户创建的审核数据。如果您希望为所有用户导出审核数据，请勿指定此值。
- `nDays`（可选）标识要导出持续多少天的审核记录。默认为七天。可能的值为：`all`（导出过去 365 天可用的审核数据）、`1`、`2`、`7`、`30`、`60` 和 `180`。如果要针对某个日期范围生成报表，则不要指定此值。
- `startDate` 和 `enddate`（可选）标识要导出审核记录的开始和结束日期。必须以 *YYYY-MM-DD* 格式指定这些值。
  - 开始日期必须早于结束日期。必须指定开始日期和结束日期才能生成报表。
  - 这些日期不应是将来的日期。
  - 如果指定了 `nDays` 可选值，将忽略这些日期
- `excludeApplicationId`（可选）标识是否要将应用程序标识符写入导出文件。默认值为 `false`。

> **注：**
>
> 不包含应用程序标识符的导出文件中的数据无法导入到 Oracle Fusion Cloud Enterprise Performance Management 环境。

示例

- 导出指定用户过去 30 天内具有应用程序标识符的审核数据：
  ```
  epmautomate exportAppAudit auditData userNames=johnDoe,jane.doe@example.com ndays=30
  ```
- 导出指定用户过去 30 天内没有应用程序标识符的审核数据：
  ```
  epmautomate exportAppAudit auditData userNames=johnDoe,jane.doe@example.com ndays=30 excludeApplicationId=true
  ```
- 导出所有用户 2024 年 8 月具有应用程序标识符的审核数据：
  ```
  epmautomate exportAppAudit auditData startDate=2024-08-01 endDate=2024-08-31
  ```

exportAppSecurity

将对象级别的访问权限分配 (ACL) 导出到 CSV 文件，您可以下载此文件以存储到本地。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate exportAppSecurity EXPORT_FILE_NAME.CSV`，其中 *EXPORT_FILE_NAME* 是将存储所导出安全数据的文件的名称。将在发件箱中创建此文件，您可以从发件箱将此文件下载到计算机。

示例

```
epmautomate exportAppSecurity app_security.CSV
```

exportARApplicationProperties

将 Account Reconciliation 应用程序设置（与 Redwood 体验、主题、电子邮件通知和业务流程名称相关）、背景图像和标识图像导出到 JSON 文件，以便您可以将它们导入同一环境或另一个环境。

当您将应用程序从生产环境导入到测试环境时，此命令很有用。如果您的应用程序设置在生产和测试环境中不同，您可以先从测试环境中导出它们，再从生产环境导入应用程序，然后将设置导入到测试环境以保持原始设置。

适用于

Account Reconciliation

所需角色

服务管理员

用法

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

- *FILE_NAME* 是将存储所导出属性值的 JSON 文件的名称。
  可以使用 downloadFile 命令下载导出文件。使用 uploadFile 命令将其上传到目标环境，然后运行 importARApplicationProperties 命令在目标环境中还原这些设置。
- `Properties`（可选）是要导出的属性的逗号分隔列表。可以导出以下部分或全部属性。如果省略此属性，则会导出所有这些属性：
  - `Theme`：导出环境中使用的显示主题。
  - `EmailNotification`：导出环境中定义的电子邮件通知设置。
  - `DisplayBusinessProcessName`：导出是否在环境中的页面上显示业务流程名称。
  - `RedwoodExperience`：导出环境的 Redwood 体验设置。
  - `BackgroundImage`：导出环境中使用的背景图像
  - `LogoImage`：导出环境中使用的标识映像。

示例

仅从环境中导出电子邮件通知和 Redwood 体验设置以及标识图像：

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

exportBackgroundImage

将 Account Reconciliation 环境中使用的背景图像导出到 JPG 文件，以便您可以将其导入其他环境。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmautomate exportBackgroundImage IMAGE_NAME.jpg`，其中 *IMAGE_NAME* 是背景图像文件的名称。
可以使用 downloadFile 命令下载图像文件。使用 uploadFile 命令将其上传到目标环境，然后运行 importBackgroundImage 命令导入该文件。

示例

```
epmautomate exportBackgroundImage corpImage.jpg
```

exportCellLevelSecurity

将单元格级别安全性设置从业务流程导出到一个 ZIP 文件中，然后可以使用 downloadFile 命令将该文件下载到本地计算机上。

适用于

Planning、Planning 模块、自由形式、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

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

- `FILE_NAME` 是将创建的 ZIP 文件的名称，该文件用于存放包含单元格级别安全性信息的 Excel 文件。
- `names`（可选）标识应用程序中单元格级别安全定义的逗号分隔列表。如果未提供此选项，则会导出应用程序中所有的单元格级别安全定义。

示例

- **导出特定的单元格级别安全定义**
  ```
  epmautomate exportCellLevelSecurity ExportCLSDRecordsFile.zip
  names=CLSDAccountPeriod,CLSDEntityPeriod,CLSDProductPeriod
  ```
- **导出所有的单元格级别安全定义**
  ```
  epmautomate exportCellLevelSecurity ExportCLSDRecordsFile.zip
  ```

exportConsolidationJournals

使用 Financial Consolidation and Close 中定义的作业导出合并日记帐。

适用于

Financial Consolidation and Close

所需角色

服务管理员

用法

`epmautomate exportConsolidationJournals jobName [fileName=FILE_NAME]`，其中

- `jobName` 是在 Financial Consolidation and Close 中创建的“日记帐导出”作业的名称。
- `fileName`（可选）是日记帐分录要导出到的 .JLF 文件的名称。使用 downloadFile 命令将此文件下载到本地计算机。

示例

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

exportData

使用在 `export data` 类型的作业中指定的导出数据设置（包括文件名），将应用程序数据导出到 ZIP 文件中。

导出的数据文件存储在默认下载位置，您可以从该位置将该文件下载到计算机。可使用收件箱/发件箱资源管理器查看导出的文件的详细信息。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate exportData JOB_NAME [FILE_NAME]`，其中 *JOB_NAME* 是在应用程序中定义的作业的名称，*FILE_NAME* 是要将数据导出到的 ZIP 文件（可选）的名称。

示例

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

exportDataManagement

将数据管理记录从环境导出到 ZIP 文件。

此命令将一组完整的设置和临时表数据（包括 ID 列）导出到 ZIP 文件，以便正确导入这些数据而不丧失引用完整性。

例如，导出的文件 `dataFile.zip` 存储在发件箱中。您可以使用 downloadFile 命令（例如 `epmAutomate downloadFile outbox/dataFile.zip`）下载导出的文件。可以使用此 ZIP 文件，通过 importDataManagement 命令导入数据。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate exportDataManagement` *`FILE_NAME`*`.zip`，其中 *`FILE_NAME`* 是要将数据导出到的 ZIP 文件的名称。

示例

```
epmautomate exportDataManagement dataFile.zip
```

exportDimension

将维从 Oracle Fusion Cloud Enterprise Data Management 应用程序导出到暂存区中的文件，或（可选）导出到连接中定义的目标环境。

适用于

Oracle Enterprise Data Management Cloud

所需角色

- 服务管理员
- 用户（需要数据管理员权限）

用法

`epmautomate exportDimension` *`APPLICATION DIMENSION FILE_NAME`* `[connection=`*`NAME`*`]`，其中：

- `APPLICATION` 是 Oracle Enterprise Data Management Cloud 应用程序的名称
- `DIMENSION` 是应用程序维的名称
- `FILE_NAME` 是用于存储已导出数据的文件（CSV 用于导出到文件，ZIP 用于导出到 Oracle Financials Cloud）的名称。如果未设置 connection 参数值，则在暂存区中创建该文件。您可以使用 downloadFile 命令将该文件下载到本地计算机，或使用 copyFileFromInstance 命令将其复制到其他 Oracle Fusion Cloud Enterprise Performance Management 环境。
- `connection=`*`NAME`*（可选）标识 Oracle Enterprise Data Management Cloud 中定义的连接名称（实例位置）。如果指定，导出文件将上传到目标环境（云 EPM 的收件箱，以及 Oracle Financials Cloud 的默认上传位置）。

> **注：**
>
> 在连接定义中指定的凭据必须具有向目标环境写入的访问权限。

示例

- 导出到 Oracle Enterprise Data Management Cloud 暂存区：`epmautomate exportDimension USOperations Entity EntityData.CSV`
- 导出并上传到 Oracle Financials Cloud：`epmautomate exportDimension USOperations Entity EntityData.zip Connection=ora_fusion_gl`
- 导出并上传到目标云 EPM 收件箱：`epmautomate exportDimension USOperations Entity EntityData.CSV Connection=EPM_cloud_pln`

exportDimensionMapping

导出某个位置的特定 Oracle Fusion Cloud Enterprise Data Management 维的映射规则以创建映射规则文件，并（可选）将导出的文件上传到其他 Oracle Fusion Cloud Enterprise Performance Management 环境的数据管理收件箱。

适用于

Oracle Enterprise Data Management Cloud

所需角色

- 服务管理员
- 用户（具有数据管理员权限）

用法

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

- `APPLICATION` 是 Oracle Enterprise Data Management Cloud 应用程序的名称
- `DIMENSION` 是应用程序维的名称
- `LOCATION` 是应为其导出映射规则的特定位置。
- `FILE_NAME` 是用于存储所导出映射的 CSV 文件的名称。如果未设置 `connection` 参数，则将在暂存区中创建该文件；您可以使用 downloadFile 命令将该文件下载到本地计算机，或使用 copyFileFromInstance 命令将其复制到其他云 EPM 环境。
- `connection=`*`NAME`*（可选）标识 Oracle Enterprise Data Management Cloud 中定义的连接名称（实例位置）。如果指定，该命令将导出的文件上传到目标环境的默认上传位置。

> **注：**
>
> 在连接中指定的凭据必须具有向目标环境写入的访问权限。

示例

- 导出到暂存区：`epmautomate exportDimensionMapping USOperations Entity Loc1 Loc1Mappings.CSV`

- 导出并上传到目标云 EPM 环境的收件箱：`epmautomate exportDimensionMapping USOperations Entity Loc1 Loc1Mappings.CSV Connection=EPM_cloud_pln`

exportEJJournals

将可从 Financial Consolidation and Close 推送的企业日记帐导出到 ZIP 文件。然后，此文件可以用于将日记帐数据导入到 ERP 系统。

将日记帐导出到导出文件后，此命令将每个导出日记帐的推送状态从 `Ready To Post`（可推送）更新为 `Post In Progress`（正在推送）。

适用于

Financial Consolidation and Close

所需角色

服务管理员

用法

`epmautomate exportEJJournals` *`FILE_NAME`*`.zip [year=`*`YEAR`*`] [period=`*`PERIOD`*`] [OPERATION=posting|validation]`，其中：

- *`FILE_NAME`* 标识要将日记帐导出 CSV 文件存档到的 ZIP 文件。该命令为每个日记帐生成一个 CSV 文件（名称格式为 `YEAR_PERIOD_JOURNALID_YYYYDDMMHHMMSS.csv`），然后将其压缩以创建此 ZIP 文件。
- *`YEAR`*（可选）是要导出日记帐数据的数据收集年份。如果未指定，将导出所有年份的数据。
- *`PERIOD`*（可选）是要导出日记帐数据的数据收集期间。仅当指定数据收集年份时才能设置此项。如果未指定值，将导出所有期间的数据。

> **Note:**
>
> 如果未指定 *`YEAR`* 和 *`PERIOD`*，此命令将导出所有年份和期间内处于 `Ready To Post`（可推送）推送状态的所有日记帐。

- `Operation`（可选）是要执行的操作。可能的值为 `posting` 和 `validation`。默认值为 `posting`。
  `operation=posting` 提供一个包含日记帐推送结果的 CSV 文件，其中包含以下列：年、期间、日记帐 ID、推送状态（已推送/失败）、消息、错误代码和错误消息。“消息”、“错误代码”和“错误消息”列为可选列。日记帐的推送状态根据文件中提供的状态进行更新。仅更新推送状态为正在推送的日记帐。

  `operation = validation` 生成一个包含日记帐验证结果的 CSV 文件，其中包含以下列：年、期间、日记帐 ID、验证状态（有效/失败）、消息、错误代码、错误消息。“消息”、“错误代码”和“错误消息”列为可选列。日记帐的验证状态根据文件中提供的状态进行更新。仅更新验证状态为正在验证的日记帐。

示例

- 导出所有年份和期间的日记帐数据：
  `epmautomate exportEJJournals Journal_Export.zip`
- 导出特定年份的日记帐数据：
  `epmautomate exportEJJournals Journal_Export.zip year=2020`

- 导出特定年份和期间组合的日记帐数据：
  ```
  epmautomate exportEJJournals Journal_Export.zip year=2024 period=March
  ```
- 导出特定年份和期间组合的日记帐数据以执行验证操作：
  ```
  epmautomate exportEJJournals Journal_Export.zip year=2024 period=March
  Operation=validation
  ```

exportEssbaseData

将数据从应用程序多维数据集导出到存档。您可以仅导出 0 级数据（ASO 和 BSO 多维数据集），或导出多维数据集中的所有数据（BSO 多维数据集）。

使用导出的存档分析 Oracle Essbase 数据中的模式，例如帮助提高性能。

> **Note:**
>
> - 存档中包含的数据文件具有 `.txt` 扩展名。
> - 智能列表和单元格文本对象作为 UUID 导出。

运行数据导出会将多维数据集置于只读模式，并在导出操作进行时阻止任何写入活动。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate exportEssbaseData` *`CUBE_NAME FILE_NAME`* `[level=0|All]`，其中：

- *`CUBE_NAME`* 标识从中导出数据的多维数据集。
- *`FILE_NAME`* 是包含导出数据的 ZIP 文件的名称。您可以通过运行 downloadFile 命令下载此存档。
- `level`（可选）标识要导出的数据级别。默认值为 `0`。
  - **ASO 多维数据集：**指定 `0` 以导出 0 级数据。不能使用 `All` 选项。
  - **BSO 多维数据集：**指定 `0` 以导出 0 级数据，或指定 `All` 以导出所有数据。

示例

- 从 BSO 多维数据集导出所有数据：
  ```
  epmautomate exportEssbaseData Report1 Report1_all_data.zip level=All
  ```
- 从多维数据集导出 0 级数据：
  ```
  epmautomate exportEssbaseData Plan1 Plan1_lvl0_data.zip
  ```

exportJobConsole

将作业控制台记录导出到 CSV 文件并创建导出 ZIP 文件。

输出 CSV 文件中的第一个字符是字节顺序标记 (Byte Order Mark, BOM) 字符 `\ufeff`，后跟用双引号括住的加密应用程序标识符。CSV 文件头跟在应用程序标识符后面。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

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

- *`FILE_NAME`* 是将存储所导出作业控制台记录的 ZIP 文件的名称。可以使用 downloadFile 命令从环境中下载此文件。
- `nDays`（可选）标识要导出多少天的作业控制台记录。可能的值为：`all`（全部小写）导出所有可用的作业控制台记录、`1`、`2`、`7`、`30` 和 `60`。默认值为 `7`。
- `jobTypes`（可选）是应导出控制台记录的作业代码的逗号分隔列表。默认值为 `Rules`。有效值为：
  - `all`（全部小写）
  - `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（可选）是要导出记录的作业状态代码的逗号分隔列表。默认值为 2（已成功完成）。可能的值为：
  - all（全部小写）任何状态下的所有作业
  - 1 - 正在处理
  - 2 - 已成功完成
  - 3 - 失败，出现错误
  - 4 - 已完成，但处于未知状态
  - 5 - 已完成，但处于超出阈值状态
  - 6 - 待取消
  - 7 - 已取消
  - 8 - 已完成，但有错误
  - 9 - 已完成，但有警告
- *exportErrorDetails*（可选）将失败或报告错误的作业的详细信息导出到日志文件（如果此值设置为 true）。此错误日志文件包含在输出 ZIP 文件中。默认值为 false。如果此值设置为 true，将导出处于以下状态的作业的状态详细信息。
  - 失败，出现错误
  - 已完成，但处于未知状态
  - 已完成，但处于超出阈值状态
  - 已完成，但有错误
  - 已完成，但有警告
- excludeApplicationId（可选）标识是否要将应用程序标识符写入导出文件。默认值为 false。

> **Note:**
>
> 不包含应用程序标识符的导出文件中的数据无法导入到 Oracle Fusion Cloud Enterprise Performance Management 环境。

示例

- 导出所有可用的作业控制台记录：

```
epmautomate exportJobConsole jobs.zip nDays=all jobTypes=all
jobStatusCodes=all
```

- 导出所有可用的规则作业控制台记录：
  ```
  epmautomate exportJobConsole jobs.zip nDays=all jobStatusCodes=all
  ```
- 导出不带应用程序标识符的所有可用的规则作业控制台记录：
  ```
  epmautomate exportJobConsole jobs.zip nDays=all jobStatusCodes=all
  excludeApplicationId=true
  ```
- 仅导出在过去 14 天内成功完成的规则作业的记录：
  ```
  epmautomate exportJobConsole jobs.zip nDays=14
  ```
- 将导入元数据的控制台记录和错误导出，并清除在过去七天内运行但处于“失败，出现错误”或者“已完成，但有错误”状态的多维数据集作业：
  ```
  epmautomate exportJobConsole jobs.zip jobtypes=IMPORT_METADATA,CLEAR_CUBE
  jobStatusCodes=3,8 exportErrorDetails=true
  ```

exportLibraryArtifact

导出 Narrative Reporting 库对象。（可选，仅限报表对象）此命令可以将导出转换为 LCM 文件，您可以将该文件导入到 Financial Consolidation and Close、Planning、Planning 模块或 Tax Reporting。

完成导出时，使用 downloadFile 命令将导出和错误文件下载到本地计算机。

适用于

Narrative Reporting

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

必须通过 ACL 为具有超级用户、用户和查看者角色的用户授予额外的安全性

用法

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

- *`ARTIFACT_PATH`* 是 Narrative Reporting 库中对象的位置。
- *`EXPORT_FILE`* 是对象要导出到的文件的唯一名称。
- `exportFormat`（可选）是以下值之一：
  - `Native`，将对象导出为 zip 文件，该文件可以用于其他 Narrative Reporting 环境。这是默认值。
  - `File`，以对象在 Narrative Reporting 中的原始二进制格式（PDF、DOCX、Zip 和 JPEG 等）导出文件。此参数只能用于导出二进制文件；它不应该用于报表对象。
  - `LCM`，将报表转换为迁移使用的格式，并将这些报表导出到 ZIP 文件，之后可将此文件导入 Financial Consolidation and Close、Planning、Planning 模块或 Tax Reporting 环境。
- `applicationName`（可选）是计划将报表导入到的目标应用程序的名称。仅当要使用 `LCM` 作为 `exportFormat` 参数的值时才需要此值。

- *errorFile*（可选）是将存储导出相关错误的文本文件的唯一名称。

示例

- 以本机格式导出报表，以便将其导入另一个 Narrative Reporting 环境：
  ```
  epmautomate exportLibraryArtifact "Library/Samples/Sample Report 1"
  exp_SampleReport1.doc errorFile=export_errors.txt
  ```
- 以原始二进制格式导出电子表格：
  ```
  epmautomate exportLibraryArtifact "Library/Spreadsheets/Sheet1.xlsx"
  exp_Sheet1.xlsx exportFormat=File errorFile=export_errors.txt
  ```
- 导出报表并设置报表格式，以将其导入 Financial Consolidation and Close、Planning、Planning 模块或 Tax Reporting：
  ```
  epmautomate exportLibraryArtifact "Library/Samples/Sample Report 1"
  exp_SampleReport1.zip exportFormat=LCM applicationName=Vision
  errorFile=report_exp_errors.txt
  ```

exportLibraryDocument

将报表库中可用的文档导出到文件。

可以使用 downloadFile 命令将导出的文件下载到本地计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

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

- *ARTIFACT_PATH* 是报表库中文档的位置或通用唯一标识符 (UUID)。要导出多个文档，请指定以逗号分隔的 UUID 或文档位置列表。
- *jobName*（可选）是要用于导出文档的库文档导出作业的名称。默认作业名称为 `Copy Artifact From Library`（从库复制对象）。
- *EXPORT_FILE*（可选）是导出文件的唯一名称。
  如果您使用 `exportFormat=ZIP`，则生成的 ZIP 文件将使用此名称。确保在此值中使用 .zip 扩展名。

  如果您使用 `exportFormat=file`：
  - 导出一个文档：如果您指定的文件扩展名与文档的扩展名匹配，则以此名称创建导出文件。
  - 导出多个文档：该命令忽略此值，并使用其原始名称导出每个文档。
- `exportFormat`（可选）是以下值之一：
  - `File`：以文档在库中的原始二进制格式（PDF、DOCX、Zip、JPEG 等）导出文档。每个导出的文件均使用其原始名称在发件箱中创建。这是默认值。

  - `zip`：创建包含原始二进制格式的所有导出文档的 ZIP 文件。如果没有为 `exportFile` 可选参数指定任何值，则使用运行命令的用户的用户名和时间戳自动生成 zip 文件名。文件名格式为 *username_exported_artifacts_yyyyMMddHHmm.zip*，例如 *JDoe_exported_artifacts_202410201559.zip*。
- *`errorFile`*（可选）是将存储导出相关错误的文件的唯一名称。如果未指定此值，则将不创建错误文件。
- `overwrite`（可选）控制是否应覆盖默认下载位置中当前存在的名称相同的文件。默认值为 `false`，这意味着，如果发件箱中存在同名文件，命令将失败。

示例

- 导出一个文档：
```
epmautomate exportLibraryDocument Library/folder1/WeeklySales.html
jobName="Copy Weekly Sales" exportFile=WeeklySales.zip
errorFile=WeeklySalesError.log overWrite=true exportFormat=zip
```
- 导出多个文档
```
epmautomate exportLibraryDocument Library/folder1/WeeklySales.html,Library/
folder2/WeeklySalesReport.pdf jobName="Copy Weekly Sales"
exportFile=WeeklySales.zip errorFile=WeeklySalesError.log overWrite=true
exportFormat=zip
```

exportLogoImage

将 Account Reconciliation 业务流程中使用的公司标识导出到 JPG 文件，以便您可以将其导入其他环境。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmautomate exportLogoImage` *`IMAGE_NAME`*`.jpg`，其中 *`IMAGE_NAME`* 是标识图像文件的名称。可以使用 downloadFile 命令下载导出的标识文件。使用 uploadFile 命令将其上传到目标环境，然后运行 importLogoImage 命令。

示例

```
epmautomate exportLogoImage corpLogo.jpg
```

exportMapping

导出特定维或位置的映射规则来创建映射规则文件。必须指定文件名和收件箱内的位置（例如，`inbox/exportedAccountMap.txt` 或 `inbox/france sales/exportedAccountMap.txt`）才能导出映射。

使用 downloadFile 命令将导出的映射文件下载到本地计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate exportMapping` *`DIMENSION_NAME|ALL FILE_NAME LOCATION`*，其中：

- *`DIMENSION_NAME|ALL`* 是从中导出映射的源维。指定要从中导出映射的维的名称，或指定 `ALL` 从某个位置的所有维导出映射。
- *`FILE_NAME`* 是映射文件的唯一名称和收件箱内的位置。
- *`LOCATION`* 是应导出映射规则的数据管理位置。

示例

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

exportMetadata

使用在 `export metadata` 类型的作业中指定的设置，将元数据导出到文件中。包含导出的数据的文件存储在默认下载位置，您可以从该位置将该文件下载到本地计算机。

（可选）您可以指定导出数据的文件名，它将覆盖默认文件名（导出元数据作业的名称）。元数据仅导出为 ZIP 文件。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate exportMetadata` *`JOB_NAME`* `[`*`FILE_NAME`*`]`，其中 *`JOB_NAME`* 是在应用程序中定义的作业的名称，*`FILE_NAME`* 是要将元数据导出到的 ZIP 文件的名称。
使用 downloadFile 命令将此文件下载到本地服务器。

示例

`epmautomate exportMetadata dailyAccountexport Accountexport.ZIP`

exportOwnershipData

将所有权数据从实体导出到逗号分隔 CSV 文件。

由 Financial Consolidation and Close 填充的默认所有权数据不包括在导出文件中。只有用户输入的用于覆盖默认设置的数据会包括在导出文件中。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 用户

用法

`epmautomate exportOwnershipData` *`Entity Scenario Year Period FILE_NAME`*，其中：

- `Entity` 是要从中导出数据的实体的名称。
- `Scenario` 是要从中导出数据的方案。
- `Year` 是要从中导出数据的年份。
- `Period` 是要从中导出数据的年份的期间。
- `FILE_NAME` 是要将数据导出到的 CSV 文件的名称。使用 downloadFile 命令将此文件下载到本地服务器。

示例

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

exportQueryResults

运行应用程序中定义的查询并将结果导出到文本文件中。

查询结果文件存储在 profitoutbox；可以使用 downloadFile 命令或者使用 Profitability and Cost Management 文件资源管理器下载该文件。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

`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]`，其中：

- *`APPLICATION_NAME`* 是要为其运行查询的 Profitability and Cost Management 应用程序的名称。
- `fileName` 是将存储查询结果的文件的名称。在未指定 `queryName` 参数值时需要提供此参数值。在指定 `queryName` 参数值时此参数是可选的，此时查询名称用作查询结果文件的名称。您指定的数据格式用于确定输出文件的格式。如果使用 `dataFormat=NATIVE`（默认值），则导出过程将创建文本文件。如果使用 `dataFormat=COLUMNAR`，则导出过程将创建多个按顺序编号的文本文件，并将它们压缩到 Zip 文件中。
- `fileOutputOptions`（可选）标识查询结果文件的输出格式。默认值为 `ZIP_ONLY`，它根据是否为 `fileName` 参数指定了值来创建 `fileName`.ZIP 或 `queryName`.ZIP。其他选项为 `TEXT_ONLY`（创建文本文件形式的输出文件）和 `ZIP_AND_TEXT`（生成文本文件和 zip 文件）。
- `queryName` 是可选参数，标识应用程序中定义的查询。包含空格字符的查询名称必须用双引号括起来。
  如果要导出属于应用程序的所有 Oracle Essbase 数据，则不要指定查询名称。

  以下情况可能会导致此命令创建空数据文件：
  - 不检索任何数据的格式错误的查询
  - 生成太多数据的查询。对于此情况，请考虑缩小查询范围从而其检索较少数据，或者将该查询分为更小的查询
    请参阅《管理 *Profitability and Cost Management*》中的“管理 Oracle Profitability and Cost Management Cloud 查询”。
- `exportOnlyLevel0Flg`（可选）指定查询是否只应检索 0 级数据。请以全部小写形式指定此参数值。
  如果您通过省略查询命令来导出所有应用程序数据，将忽略此参数。
- `roundingPrecision`（可选），指定要在导出查询结果时使用的小数位数（舍入精度）。仅在指定 `queryName` 时可用。默认值为 2。
- `dataFormat`（可选）标识输出格式。有效值为：
  - `NATIVE`，将查询结果作为 Essbase 本机格式数据进行维护。这是默认值。
  - `COLUMNAR`，转换 Essbase 本机格式数据并在列中对数据进行排序，以便于解释和导入到其他应用程序。
    此选项将导出所有 Essbase 数据并忽略 `queryName` 参数值。可以通过设置 `memberFilters` 参数值筛选数据。

  > **注：**
  >
  > 仅当 `dataFormat` 指定为 `COLUMNAR` 时，该命令才考虑以下可选参数。

- `memberFilters`（可选）接受 JSON 格式的字符串，以按维和 0 级成员进行筛选。例如，`"{\"Dim1\":[\"Mem1\"],\"Dim2\":[\"Mem21\",\"Mem22\"]}"`

- `includeHeader`（可选）添加维名称作为列标题。将此值设置为 `false` 可排除列标题。默认值为 `true`。
- `delimiter`（可选）标识在查询结果文件中用于分隔维成员的分隔符。分隔符必须用双引号括起来。默认值为空格 (" ")。
- `keepDuplicateMemberFormat`（可选）指定是否以 Essbase 重复成员格式（例如，`[Account]@[Accoun1]`）输出成员格式。将此值设置为 `false` 可仅输出成员名称。默认值为 `true`。

示例

- 导出所有应用程序数据：
```
epmautomate exportQueryResults BksML12 fileName="BksML12_MyQuery1.txt"
fileOutputOptions=TEXT_ONLY
```
- 导出特定查询的结果：
```
epmautomate exportQueryResults BksML12 queryName="My Product Query"
roundingPrecision=3
```
- 以 NATIVE 数据格式导出 0 级数据：
```
epmautomate exportQueryResults BksML30 fileName="BksML30_ExportLevel0-Data"
fileOutputOptions=ZIP_AND_TEXT exportOnlyLevel0Flg=true
```
- 使用单个维和单个成员筛选器，以 COLUMNAR 数据格式导出 0 级数据：
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Period\":[\"December\"]}"
includeHeader="true" delimiter="," roundingPrecision="3"
```
- 使用单个维和多个成员筛选器，以 COLUMNAR 数据格式导出 0 级数据：
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Period\":[\"November\",\"December\"]}"
includeHeader="true" delimiter="," roundingPrecision="3"
```
- 使用多个维和多个成员筛选器，以 COLUMNAR 数据格式导出 0 级数据：
```
epmautomate exportQueryResults BksML30 fileName="BksML30_Level0-Data"
dataFormat="COLUMNAR" memberFilters="{\"Year\":[\"2016\"],\"Period\":
[\"November\",\"December\"]}" includeHeader="true" delimiter=","
roundingPrecision="3"
```

exportSnapshot

重复之前执行的导出操作以创建迁移内容的快照。

使用迁移选择所需的对象并将其导出到快照；例如 `January16FullApp`。将快照名称与此命令一起使用，可在随后重复导出操作，这将仅导出初始导出操作期间选择的对象。请参阅《管理迁移》中的“导出对象和应用程序”。

- 以下内容不属于 Planning、Planning 模块和自由形式应用程序快照的一部分：
  - 审核数据
  - 作业控制台数据

  如果要将审核和作业控制台数据复制到目标环境，请使用 cloneEnvironment 命令或克隆环境功能。
- 快照不包含数据管理临时表数据。要导入此数据，请使用 exportDataManagement 和 importDataManagement 命令或数据管理系统维护脚本界面。您可以使用 cloneEnvironment 命令或克隆环境功能来创建环境的相同副本，包括数据管理临时表数据。

可以使用 downloadFile 命令从默认位置下载导出的快照。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate exportSnapshot` *`SNAPSHOT_NAME`*，其中 *`SNAPSHOT_NAME`* 是迁移中现有快照的名称。该快照将由新快照取代。

示例

```
epmautomate exportSnapshot January16FullApp
```

exportTemplate

将应用程序作为模板导出为 .ZIP 文件。导出的文件存储在 profitoutbox 中。

可以使用 downloadFile 命令将导出的文件下载到本地计算机。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate exportTemplate` *`APPLICATION_NAME File_Name`*，其中：

- *`APPLICATION_NAME`* 是要导出为模板的 Profitability and Cost Management 应用程序的名称
- `File_Name` 是模板文件的名称

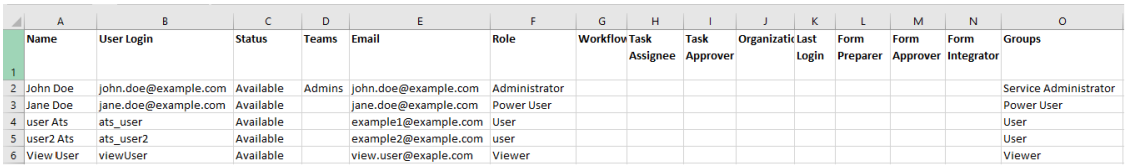
示例

```
epmautomate exportTemplate BksML12 template1
```

exportTaskManagerAccessControl

导出任务管理器、补充数据和企业日记帐用户分配的用户详细信息报表。该报表包含有关在环境中具有预定义角色的用户的信息，并将每个用户的属性（例如名称和电子邮件）以及其状态、团队、预定义角色、工作流角色、组织、组和上次登录时间戳列出到 Excel 或 CSV 文件中。

示例任务管理器访问控制报表：



	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

适用于

Enterprise Profitability and Cost Management、Financial Consolidation and Close 和 Tax Reporting

所需角色

服务管理员

用法

`epmAutomate exportTaskManagerAccessControl` *`REPORT_NAME`*，其中 *`REPORT_NAME`* 是将包含报表的导出文件的名称（包括有效的（CSV 或 XLS）扩展名）。

可以生成 CSV 或 XSL 格式的此报表。可以使用 downloadFile 命令对其进行下载。

示例

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

exportValidIntersections

将有效交叉点组从业务流程导出到一个 ZIP 文件中，然后可以使用 downloadFile 命令将该文件下载到本地计算机上。有效交叉点是基于您定义的规则（称为“有效交叉点规则”）来筛选的单元格交叉点。当用户输入数据或选择运行时提示时，这些规则将为用户筛选特定的单元格交叉点。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate exportValidIntersections` *`FILE_NAME`*`.zip [names=`*`INTERSECTION_NAMES`*`]`，其中：

- *`FILE_NAME`* 是导出 ZIP 文件的名称。在命令中标识的所有有效交叉点都先导出到一个 Microsoft Excel 文件，然后进行压缩以创建此 ZIP 文件。
- `names`（可选）标识要导出的有效交叉点的逗号分隔列表。如果未指定此参数值，则该命令将导出应用程序中的所有有效交叉点。

示例

- **导出特定的有效交叉点**

```
epmautomate exportValidIntersections VI_export_File.zip
names=VIAccountPeriod,VIEntityPeriod,VIProductPeriod
```

- **导出所有的有效交叉点**
  ```
  epmautomate exportValidIntersections VI_export_File.zip
  ```

extractDimension

将 Oracle Fusion Cloud Enterprise Data Management 维导出到文件或全局连接。

适用于

Oracle Enterprise Data Management Cloud

所需角色

- 服务管理员
- 用户（具有数据管理员权限）

用法

`epmautomate extractDimension` *`APPLICATION DIMENSION EXTRACT_PROFILE FILE_NAME`* `[connection=`*`NAME`*`]`，其中：

- `APPLICATION` 是 Oracle Enterprise Data Management Cloud 应用程序的名称。
- `DIMENSION` 是要提取的维的名称。
- `EXTRACT_PROFILE` 是在应用程序中定义的提取配置文件的名称。此配置文件用于提取维。
- `FILE_NAME` 是用于存储提取的数据的文件名（CSV 用于导出到文件，ZIP 用于导出到 Oracle Financials Cloud）。如果未设置连接参数值，则在暂存区中创建该文件。您可以使用 downloadFile 命令将该文件下载到本地计算机，或使用 copyFileFromInstance 命令将其复制到 Oracle Enterprise Data Management Cloud 环境。
- `connection=`*`NAME`* （可选）将 Oracle Enterprise Data Management Cloud 中定义的全局连接名称（实例位置）标识为文件的位置。如果指定，则提取文件将上传到目标环境（Oracle Fusion Cloud Enterprise Performance Management 的收件箱，以及 Oracle ERP 的指定文档帐户）。

> **Note:**
> 在全局连接中指定的凭据必须具有向目标环境写入的访问权限。

示例

- 提取到 Oracle Enterprise Data Management Cloud 暂存区：`epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.CSV`
- 提取并上传到 Oracle ERP：`epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.zip Connection=ora_fusion_gl`
- 提取并上传到目标云 EPM 收件箱：`epmautomate extractDimension USOperations Entity EntityExtProfile EntityData.CSV Connection=EPM_cloud_pln`

extractPackage

使用单个操作运行由应用程序的多个提取组成的提取包。包中每个提取的结果都添加到一个 ZIP 文件中，该文件可以写入暂存区或全局连接。

适用于

Oracle Fusion Cloud Enterprise Data Management

所需角色

- 服务管理员
- 用户（具有数据管理员权限）

用法

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

- *APPLICATION_NAME* 是 Oracle Enterprise Data Management Cloud 应用程序的名称。
- *EXTRACT_PACKAGE_NAME* 是 Oracle Enterprise Data Management Cloud 应用程序中定义的包的名称。
- *FILE_NAME* 是将包含提取结果的提取 ZIP 文件的名称。此文件将在暂存区中创建，如果指定了连接名称，则将其写入全局连接。
  一个包可能包含一个或多个提取配置文件，每个配置文件都有不同的名称。提取包 ZIP 文件将包含包中提取的输出文件。ZIP 文件中的文件数量由包中的设置决定。
- *CONNECTION_NAME*（可选）是 Oracle Enterprise Data Management Cloud 中定义的全局连接名称（实例位置）。它标识包含提取结果的 ZIP 文件要上传到的位置。
  如果指定连接名称，则包含提取的数据的 ZIP 文件将上传到目标环境（Oracle Fusion Cloud Enterprise Performance Management 环境的收件箱，或 Oracle ERP 的指定文档帐户）。
  如果未指定值，则会在 Oracle Enterprise Data Management Cloud 环境的默认发件箱（暂存区）中创建导出文件。

示例

- 将 COARedesignMaps 包提取到 Oracle Enterprise Data Management Cloud 暂存区中的文件：

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

- 将 COARedesignMaps 包提取到文件中，然后使用连接将其上传到 Oracle Fusion Cloud EPM 收件箱。

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

feedback

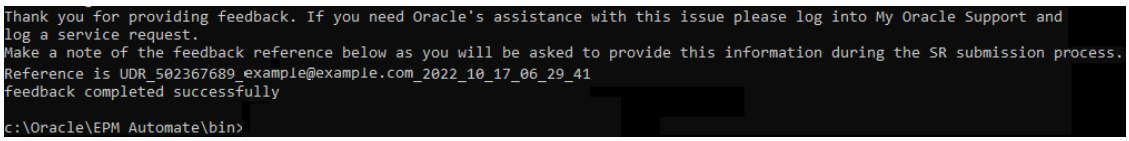
向 Oracle 和环境的服务管理员发送反馈，并自动从当前目录上传过去 24 小时内创建的所有 EPM Automate 日志文件。

您还可以选择上传要让 Oracle 技术支持用于诊断当前问题发生原因的其他文件（例如 Fiddler 跟踪文件）。

此命令模拟服务的“提供反馈”功能，在向 Oracle 提供反馈时非常有用，尤其是在用户界面无响应或运行 EPM Automate 出现问题时。

有关“提供反馈”功能的信息，请参阅《管理员入门指南》中的“使用提供反馈实用程序帮助 Oracle 收集诊断信息”。

此命令返回类似以下内容的消息，告知用户 feedback 命令未创建服务请求。如果您需要 Oracle 的帮助来解决问题，您必须填报服务请求。此命令显示 UDR 参考号；您应在填报的服务请求中包含此参考号。



适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

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

- `Comment` 是用于描述当前所提交反馈涉及的问题的文本。注释必须用引号括起来。
- `Screenshot`（可选）标识用于阐明当前所提交反馈涉及的问题的图形文件名称。可以根据需要重复此参数和值来提交多个屏幕截图。
- `File`（可选）标识要让 Oracle 技术支持用于解决当前问题的文件的名称。使用此参数将 Fiddler 跟踪文件或其他文件提交给 Oracle。可以根据需要重复此参数和值来提交多个文件。

示例

- 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

检查应用程序是否处于管理模式，并且只允许服务管理员访问。

此命令用于在运行自动化脚本之前检查应用程序的状态；如果应用程序处于管理模式，则命令返回 `true`，否则返回 `false`。例如，refreshCube 命令要求应用程序处于管理模式。您可以按以下所示在自动化脚本中使用此命令，以检查应用程序是否处于管理模式。

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

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、Account Reconciliation、战略性人员规划和销售规划。

所需角色

服务管理员

用法

```
epmautomate getApplicationAdminMode
```

示例

```
epmautomate getApplicationAdminMode
```

getDailyMaintenanceStartTime

在控制台中显示安排对环境开始日常维护的协调世界时 (UTC) 或（可选）时区。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员、用户（仅 Oracle Enterprise Data Management Cloud）、任何预定义角色和“迁移 - 管理”应用程序角色

用法

`epmautomate getDailyMaintenanceStartTime [timezone=true|false]`，其中 `timezone=true`（可选）标识是否以设置日常维护开始时间时指定的时区显示该时间，例如 `America/Los_Angeles`。默认值为 `false`。

示例

- 以设置维护时间时指定的时区显示该时间：
  `epmautomate getDailyMaintenanceStartTime timezone=true`
- 以 UTC 格式显示维护时间：
  `epmautomate getDailyMaintenanceStartTime`

getEssbaseQryGovExecTime

显示 Oracle Essbase 查询在终止之前当前可用于检索和传送信息的最长时间（以秒为单位）。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Profitability and Cost Management、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

```
epmautomate getEssbaseQryGovExecTime
```

示例命令输出：

```
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
```

示例

```
epmautomate getEssbaseQryGovExecTime
```

getIdleSessionTimeout

显示 Oracle Fusion Cloud Enterprise Performance Management 环境的会话超时（以分钟为单位）。在会话空闲了此持续时间后，用户将被重定向到登录页。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

```
epmautomate getIdleSessionTimeout
```

示例命令输出：

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

getIPAllowlist

对于 OCI（第 2 代）环境，显示当前允许列表中包含的 IP 地址和无类别域间路由 (Classless Inter-Domain Routing, CIDR)。

此命令可用于检查当前是否允许特定 IP 地址或 CIDR 访问 OCI（第 2 代）环境。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

```
epmAutomate getIPAllowlist
```

> **Note:**
>
> 要将所有现有 IP 地址和 CIDR 写入文件，请将输出重定向到文本文件，您可以对其进行编辑（删除一些或所有 IP 地址和 CIDR），上传到环境，然后使用 setIPAllowlist 命令从允许列表中删除文件中的条目。命令执行示例：
>
> ```
> epmAutomate getIPAllowlist > myRemoveList.txt
> epmAutomate uploadFile myRemoveList.txt
> epmAutomate setIPAllowlist remove myRemoveList.txt
> ```

示例

显示当前允许列表中包含的 IP 地址和 CIDR：

```
epmAutomate getIPAllowlist
```

getRestrictedDataAccess

显示环境是否配置为阻止服务管理员在使用“提供反馈”实用程序时同意向 Oracle 提交应用程序快照。

如果环境配置为阻止服务管理员同意提交应用程序快照，则此命令报告 `true`，否则报告 `false`

适用于

Account Reconciliation、Oracle Fusion Cloud Enterprise Data Management、Enterprise Profitability and Cost Management、Financial Consolidation and Close、自由形式、Planning、Planning 模块、Profitability and Cost Management、Narrative Reporting、销售规划、战略性人员规划和 Tax Reporting。

所需角色

服务管理员

用法

要更改当前数据访问限制状态，请使用 setRestrictedDataAccess。

示例

```
epmautomate getRestrictedDataAccess
```

getSubstVar

检索替代变量的值并显示在屏幕上。

显示格式为 *CUBE_NAME*.*SUBSTVAR*=value，例如，Plan2.CurYear=2016。以 *ALL*.*SUBSTVAR*=value 格式显示应用程序级别的替代变量值，例如 ALL.CurYear=2016

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员、超级用户（具有规则启动访问权限）

用法

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

- *CUBE_NAME* 是要从中检索替代变量的多维数据集（例如，Plan1、Plan2）。使用 ALL 在应用程序级别检索替代变量。
- *name=VARIABLE_NAME* 标识要检索值的替代变量（可选）。如果不指定变量名称，命令将检索所有替代变量的值。

示例

- 获取应用程序和多维数据集级别的所有替代变量的值：epmautomate getSubstVar ALL
- 在应用程序级别获取一个特定替代变量的值：epmautomate getSubstVar ALL name=CurYear
- 在多维数据集级别获取所有替代变量的值：epmautomate getSubstVar Plan2
- 在多维数据集级别获取特定替代变量的值：epmautomate getSubstVar Plan2 name=CurYear

getVirusScanOnFileUploads

检查是否允许 OCI（第 2 代）环境扫描所有要上传的文件以确保它们无病毒。

此命令检查在文件上传到环境之前是否强制执行病毒扫描。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

```
epmautomate getVirusScanOnFileUploads
```

如果为环境启用了扫描上传文件中的病毒，则此命令输出 `true`，否则输出 `false`。

groupAssignmentAuditReport

创建报表，其中列出在指定日期范围内分配到访问控制组或从中取消分配（删除）的用户和组。此报表最多可生成 120 天的数据。

此报表生成为 CSV 文件，可用于支持安全审核操作。生成的 CSV 文件的每一行都提供了添加或删除的用户或组、从中添加或删除用户或组的组、执行操作的服务管理员以及完成操作的日期和时间。

	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

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmAutomate groupAssignmentAuditReport` *`FROM_DATE TO_DATE REPORT_NAME`*，其中

- *`FROM_DATE`* 是要生成报表的期间的开始日期，以 `YYYY-MM-DD` 格式表示。
- *`TO_DATE`* 是要生成报表的期间的结束日期，以 `YYYY-MM-DD` 格式表示。
- *`REPORT_NAME`* 是报表的 CSV 文件的名称。可以使用 downloadFile 命令下载生成的报表。

示例

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

help

显示所有 EPM Automate 命令或特定命令的帮助。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

`epmautomate [`*`COMMAND_NAME`*`] help`，其中 *`COMMAND_NAME`*（可选）是要为其在控制台中显示帮助的命令的名称

示例

- 显示可从中访问所有命令的帮助的网页：
  ```
  epmautomate help
  ```
- 在控制台中显示有关加密命令的命令帮助：
  ```
  epmautomate encrypt help
  ```

importAppAudit

使用您通过导出环境中的审核数据所创建的 ZIP 文件，导入数据审核记录。

可使用 exportAppAudit 命令 (`epmautomate exportAppAudit auditData ndays=All`) 创建导入文件。在进行迁移或为灾难恢复进行克隆期间，可以使用此命令将审核记录从一个环境克隆到另一个环境。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate importAppAudit FILE_NAME [logFilename=LOG_FILE_NAME]`，其中：

- `FILE_NAME` 是 ZIP 文件的名称，该文件中包含要导入到应用程序的数据审核记录。在运行此命令之前，请先使用 uploadFile 命令将此文件上传到环境。
- `logFileName`（可选）标识将记录导入过程中遇到的错误的错误日志文件。如果未指定此值，该命令将生成使用以下惯例命名的错误文件：`username_date_timestamp`。可以使用 downloadFile 命令下载此文件。

示例

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

importAppSecurity

从收件箱中可用的 CSV 文件为应用程序的用户或组加载访问权限。

导入访问权限操作将只覆盖对以下各项的现有分配：导入的成员、数据表单、数据表单文件夹、任务列表、Calculation Manager 业务规则，以及业务规则文件夹。所有其他现有访问权限都将保持不变。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

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

- `ACL_FILE_NAME` 是 CSV 文件的名称，该文件中包含要导入到应用程序的访问权限。运行此命令前，请使用 uploadFile 命令将此文件上传到收件箱。示例输入文件的内容可能如下图所示：

	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

有关列标题和可能值的说明，请参阅《*REST APIs*》指南中的 "Import Security"。

- `ERROR_FILE` 是 CSV 文件的名称，EPM Automate 在发件箱中创建此文件，以记录在此操作期间检测到的错误。您可以将此文件下载到本地计算机，以分析和纠正报告的错误。示例错误文件的内容可能如下图所示：此文件的列与输入文件的标题列相对应：

	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`（可选）指定在从文件加载新权限前是否删除现有访问权限。默认值为 `false`。

示例

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

importARApplicationProperties

将导出 JSON 文件中可用的应用程序设置（Redwood 体验、主题、电子邮件通知和业务流程名称）、标识和背景图像导入到 Account Reconciliation 环境。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmautomate importARApplicationProperties` *`FILE_NAME`*，其中 *`FILE_NAME`* 是从环境导出的 JSON 文件的名称。

使用 exportARApplicationProperties 命令从其他环境导出的此文件必须在还原应用程序设置的环境中可用。

示例

```
epmautomate importARApplicationProperties myProp.JSON
```

importBackgroundImage

将背景图像从导出文件导入到 Account Reconciliation 环境。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmautomate importBackgroundImage` *`FILE_NAME`*`.jpg`，其中 *`FILE_NAME`* 是从其他环境导出的背景图像文件的名称。

示例

```
epmautomate importBackgroundImage image_file.jpg
```

importBalances

使用数据管理根据数据加载定义导入余额数据。

适用于

Account Reconciliation。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

必须通过 ACL 为具有超级用户、用户和查看者角色的用户授予额外的安全性

用法

`epmautomate importBalances DL_DEFINITION PERIOD`，其中：

- *DL_DEFINITION* 是 Account Reconciliation 中现有的数据加载定义。
- *PERIOD* 是期间的名称。

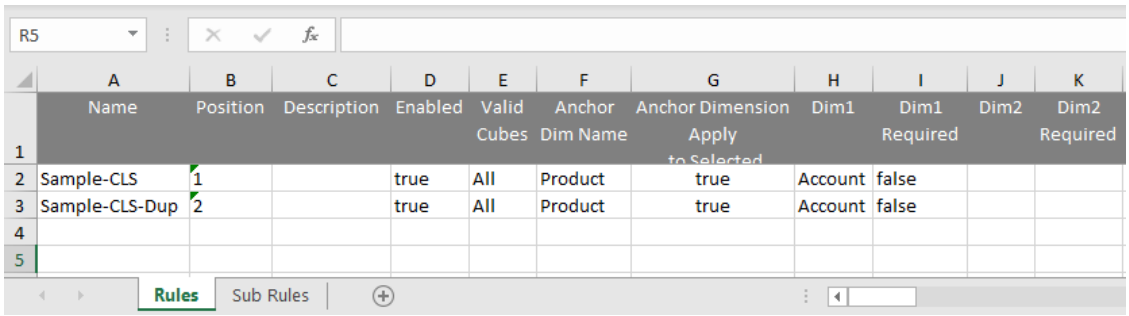
示例

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

importCellLevelSecurity

将单元格级别安全性设置从 ZIP 文件导入业务流程中，该 ZIP 文件含有一个包括单元格级别安全性记录的 Excel 文件。在运行此命令之前，请先使用 uploadFile 命令将导入文件上传到环境。

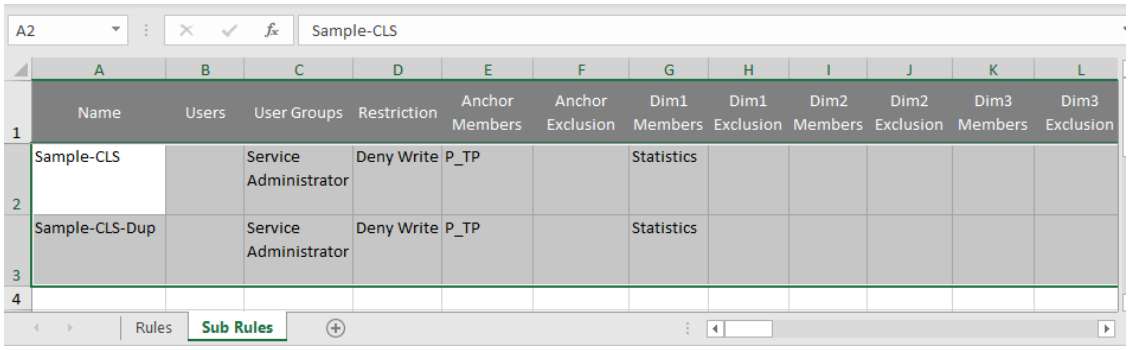
您的导入 ZIP 文件应当包含一个 Excel 文件，该 Excel 文件应具有两个工作表（Rules 工作表和 Sub Rules 工作表）才能成功导入单元格级别安全性。Rules 表应当包含单元格级别安全定义、所包括的维、属性（如“未指定有效”和“需要的其他维”）Sub Rules 表应当包含成员选择和排除。获取导入文件格式模板的最佳方式是从应用程序中导出单元格级别安全性。下图中显示了示例格式。



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



适用于

Planning、Planning 模块、自由形式、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

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

- `FILE_NAME` 是 ZIP 文件的名称，该 ZIP 文件中含有一个包括单元格级别安全性信息的 Excel 文件。
- `ErrorFile`（可选）标识要向其中写入错误记录的文本文件的名称。如果未指定此参数值，该命令会自动生成一个错误文件；您可以在作业控制台中查看该错误文件的名称。使用 downloadFile 命令将错误文件下载到本地计算机。

示例

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

importConsolidationJournals

将 .JLF 文件中的合并日记帐导入 Financial Consolidation and Close 中。

- 使用 exportConsolidationJournals 命令创建 .JLF 文件，该文件用作此命令的输入。
- 运行此命令前，请使用 uploadFile 命令将输入文件加载到环境中。

适用于

Financial Consolidation and Close

所需角色

服务管理员

用法

`epmautomate importConsolidationJournals jobName [fileName=`*`FILE_NAME`*`]`
`[errorFileName=`*`ERROR_FILE_NAME`*`]`，其中

- `jobName` 是在 Financial Consolidation and Close 中创建的“导入日记帐”作业的名称。
- `fileName`（可选）是要从中导入日记帐分录的 .JLF 文件的名称。
- `errorFileName`（可选）是日志文件的名称，将在此文件中记录导入过程中生成的消息。

示例

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

importData

使用在 `import data` 类型的作业中指定的导入数据设置，将数据从文件导入应用程序中。

使用 uploadFile 命令将包含应用程序数据的文件上传到默认上传位置。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

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

- *JOB_NAME* 是在应用程序中定义的作业的名称。
- *FILE_NAME*（可选）标识要从其导入数据的 ZIP、CSV 或 TXT（Essbase 格式的数据文件）文件的名称。如果您指定一个文件名，将忽略作业中的导入文件名。
  如果作业定义为以 Essbase 格式导入数据，则 ZIP 文件必须包含 Essbase 格式的 TXT 文件。对于其他导入作业，ZIP 文件可包含一个或多个 CSV 文件，这些文件在文件名中标识了导入序列；例如 `data1-3.csv`、`data2-3.csv` 和 `data3-3.csv`。
- `errorFile`（可选）标识 ZIP 文件的名称，该文件中将记录导入操作期间被拒绝的记录（如果有）。将覆盖发件箱中名称相同的 ZIP 文件。可以使用 downloadFile 命令下载此文件。

示例

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

importDataManagement

将数据管理记录从 ZIP 文件导入到环境。

此命令将使用 exportDataManagement 命令创建的 ZIP 文件中的数据导入到设置表和临时表。服务管理员使用 uploadFile 命令（例如 `epmAutomate uploadFile "C:/datafile/datafile.zip" inbox`）将导入 ZIP 文件上传到数据管理收件箱或其中的文件夹。

> **Note:**
>
> 只有当另一个环境也基于相同的每月更新运行时，此命令才能导入从该环境导出的数据管理记录。例如，从 21.11 Oracle Fusion Cloud Enterprise Performance Management 环境中导出的记录只能导入到另一个 21.11 环境。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Account Reconciliation、Tax Reporting、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate importDataManagement` *`FILE_NAME`*`.zip`，其中 *`FILE_NAME`* 是包含要导入的数据管理数据的 ZIP 文件的名称。

示例

- 从数据管理收件箱导入：
  `epmautomate importDataManagement inbox/dataFile.zip`
- 从收件箱中的文件夹导入：
  `epmautomate importDataManagement inbox/dm_data/dataFile.zip`

importDimension

将维从文件导入到 Oracle Fusion Cloud Enterprise Data Management 应用程序中。

此命令可从 Oracle Enterprise Data Management Cloud 中定义的连接或暂存区导入输入文件。

如果将从 Oracle Enterprise Data Management Cloud 暂存区导入文件，则服务管理员使用 uploadFile 命令将文件上传到目标 Oracle Enterprise Data Management Cloud 环境。服务管理员还可以使用 copyFileFromInstance 命令从其他 Oracle Fusion Cloud Enterprise Performance Management 环境复制文件。

适用于

Oracle Enterprise Data Management Cloud。

所需角色

- 服务管理员
- 用户（具有数据管理员权限）

用法

`epmautomate importDimension` *`APPLICATION DIMENSION IMPORT_TYPE FILE_NAME`* `[connection=`*`NAME`*`]`，其中：

- `APPLICATION` 是 Oracle Enterprise Data Management Cloud 应用程序的名称

- DIMENSION 是所导入的应用程序维的名称
- IMPORT_TYPE 指示执行导入的方式。有效的导入类型为：
  - ResetDimension，用于删除所有现有维并导入新数据
  - ReplaceNodes，用于在导入过程中添加或更新节点并替换现有层次
  - Merge，用于通过使用导入请求处理节点和层次的增量更改
- FILE_NAME 是包含要导入的维数据的文件（CSV 或 ZIP）名称。文件名必须以前缀为 _（下划线字符）的维名称结尾；例如，import_Entity.csv。如果从包含多个导入文件的 ZIP 文件导入，则此命令依赖于 ZIP 文件内的文件名来确定正确的导入文件。
  如果指定 connection 的值，则必须从 ZIP 文件导入维；例如，importdata_Entity.zip。
- connection=*NAME*（可选）将 Oracle Enterprise Data Management Cloud 中定义的连接名称（实例位置）标识为导入文件的位置。如果未指定，导入过程将在本地暂存区中查找导入文件。

示例

- 从上传到暂存区的文件导入：epmautomate importDimension USOperations Entity ReplaceNodes data_Entity.CSV
- 从其他云 EPM 环境的发件箱导入：epmautomate importDimension USOperations Entity ReplaceNodes data_Entity.ZIP Connection=Cloud-EPM_pln

importJobConsole

使用 ZIP 文件克隆作业控制台记录，该文件包含从环境导出的作业控制台记录。

使用此命令导入作业控制台记录是一次性任务，应在运行 recreate 命令后执行。如果您已使用此命令导入作业控制台记录，则该命令的后续调用将失败，直到您重新创建环境。

使用 exportJobConsole 命令 (epmAutomate exportJobConsole FILE_NAME.zip nDays=All jobTypes=All jobStatusCode=All) 创建用作此命令的输入的 ZIP 文件。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

epmautomate importJobConsole *FILE_NAME*.zip [logFileName=jobConsoleLog]，其中：

- *FILE_NAME* 是包含要导入的作业控制台记录的 ZIP 文件的名称。可以使用 uploadFile 命令将此文件上传到环境。
- logFileName（可选）将 jobConsoleLog 标识为记录导入期间遇到的错误的日志文件。如果未指定此值，该命令会生成使用以下惯例命名的错误文件：*username*importLog_date_timestamp.zip。可以使用 downloadFile 命令下载此文件。

示例

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

importLibraryArtifact

将库对象从存档或文件导入 Narrative Reporting 库。

运行此命令之前，服务管理员使用上传 uploadFile 命令将源存档或文件加载到环境中。

适用于

Narrative Reporting

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

必须通过 ACL 为具有超级用户、用户和查看者角色的用户授予额外的安全性

用法

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

- *`SOURCE_FILE`* 是包含要导入到库中的对象的存档文件名称。该文件必须在收件箱中。
- *`errorFile`*（可选）是将存储导入相关错误的文本文件的唯一名称。
- `importFormat`（可选）是以下值之一：
  - `Native`，从使用带 `exportFormat=Native` 选项的 exportLibraryArtifact 命令创建的 zip 文件导入对象。这是默认值。
  - `File`，可导入二进制文件。

> **注：**
>
> 可以使用 importSnapshot 命令将库对象 zip 文件（使用带 `exportFormat=LCM` 选项的 exportLibraryArtifact 命令创建）导入到 Financial Consolidation and Close、Planning、Planning 模块或 Tax Reporting 环境。

- `importFolder`（可选）是将存储导入对象的库位置。如果此位置与 `Library`（默认导入位置）不同，则指定此路径。
- *`importPermission`* 指明是否导入对象的访问权限集。默认值为 `False`。
- *`overwrite`* 标识是否覆盖指定库位置中具有相同名称的对象（如果有）。默认值为 `False`，这意味着，如果导入位置中存在相同名称的对象，则该过程将不会导入某个对象。

完成导入时，使用 downloadFile 命令将错误文件下载到本地计算机。

示例

- 以二进制格式导入文件：

```
epmautomate importLibraryArtifact newReports.doc
errorFile=report_imp_errors.txt importFormat=File importFolder="Library/My
Reports" importPermission=true overwrite=true
```

- 从导出的 zip 文件导入对象：
  ```
  epmautomate importLibraryArtifact newReports.zip
  errorFile=report_imp_errors.txt importFormat=Native importFolder="Library/My
  Reports" importPermission=true overwrite=true
  ```
- 将报表从导出的 zip 文件导入 Financial Consolidation and Close、Planning、Planning 模块或 Tax Reporting 环境：
  ```
  epmautomate importSnapshot newReports.zip
  ```

importLogoImage

将 Account Reconciliation 环境中使用的公司标识从导出文件导入到另一个环境。

适用于

Account Reconciliation

所需角色

服务管理员

用法

`epmautomate importLogoImage` *`IMAGE_NAME`*`.jpg`，其中 *`IMAGE_NAME`* 是标识图像文件的名称。可以使用 downloadFile 命令下载导出的图像。使用 uploadFile 命令将其上传到目标环境，然后运行 importLogoImage 命令导入该图像。

示例

```
epmautomate importLogoImage corpLogo.jpg
```

importMapping

从先前上传到环境的映射导入文件来导入映射。

服务管理员使用 uploadFile 命令将文件上传到数据管理收件箱或该收件箱内的某个文件夹。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

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

- *DIMENSION_NAME|ALL* 指示映射的接收方。指定要将映射导入的维的名称，或指定 ALL 将文件中包括的所有映射导入相应维。
- *FILE_NAME* 是数据管理收件箱或该收件箱内目录中的映射导入文件的名称和位置。指定文件名（标准数据管理格式的 TXT 文件）及其路径（例如 inbox/AccountMap.txt 或 inbox/pbcs_maps/AccountMap.txt）。
- *IMPORT_MODE* 是 REPLACE 或 MERGE，前者在导入之前清除现有映射规则，后者向现有规则添加新映射规则。
- *VALIDATION_MODE* 是 TRUE 或 FALSE，前者针对应用程序验证目标成员，后者加载映射文件而不运行验证。
- *LOCATION* 是应加载映射规则的数据管理位置。

示例

- epmautomate importMapping Account inbox/AccountMap.txt MERGE FALSE "France Sales"
- epmautomate importMapping ALL "inbox/France Sales/AllMaps.txt" MERGE FALSE "France Sales"（将映射从映射导入文件加载到 France Sales 位置中的所有映射维）

importMetadata

使用在 import metadata 类型的作业中指定的导入设置，将元数据导入应用程序中。（可选）您可以指定要从其导入元数据的 ZIP 文件的名称。

使用 uploadFile 命令将包含元数据的文件上传到默认上传位置。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

epmautomate importMetadata *JOB_NAME [FILE_NAME]* errorFile=*ERROR_FILE*.zip，其中：

- *JOB_NAME* 是在应用程序中定义的作业的名称。
- *FILE_NAME*（可选）标识要从其导入元数据的 ZIP 文件的名称。如果指定，则此 ZIP 文件的内容优先于在作业中定义的文件名。ZIP 文件可包含一个或多个 CSV 文件。包含维元数据的文件名应与作业中定义的导入文件名匹配或者以 *_DIMENSIONNAME*.csv 结尾；例如，metadata_Entity.csv、metadata_HSP_Smart Lists.csv 和 metadata_Exchange Rates.csv。
- errorFile（可选）标识 ZIP 文件的名称，该文件中将记录导入操作期间被拒绝的记录（如果有）。将覆盖发件箱中名称相同的 ZIP 文件。可以使用 downloadFile 命令下载此文件。

> **注：**
>
> - 不能通过使用修改了 `old_name` 或 `unique_name` 属性的加载文件运行导入元数据作业的方式来重命名成员。成员的重命名将被忽略。
> - 使用此命令导入元数据时，不能删除属性维。
> - 只会导入在作业中设置了元数据导入的维的元数据。将忽略其他维的元数据（如果包含在 ZIP 文件中）。
>
>   如果 ZIP 文件同时满足以下两个条件，将创建模糊导入：
>
>   – Zip 包含名称与作业中定义的文件名相匹配的元数据文件
>
>   – Zip 包含名称以 *_DIMENSIONNAME*.CSV 或 *_DIMENSIONNAME*.TXT 结尾的元数据文件，其中 *DIMENSIONNAME* 是元数据要导入的维的名称。
>
> Oracle 建议 ZIP 文件包含名称与作业中引用的名称相同的元数据文件，或者包含名称以 *_DIMENSIONNAME*.CSV（或 *_DIMENSIONNAME*.TXT）结尾的文件，但不能同时包含这两种文件。例如，如果正在加载的作业引用元数据文件 `Employees_A-Z.CSV` 并将其对应到 `Employees` 维，则 ZIP 可以包括 `Employees_A-Z.CSV` 或 `New_Employees.CSV`，但不能同时包括两者。如果 ZIP 包含 `Employees_A-Z.CSV` 和 `New_Employees.CSV`，则 EPM Automate 可能选择任一文件导入，具体取决于 ZIP 中文件的顺序。`Employees_A-Z.CSV` 文件是导入的潜在匹配项，因为它的名称与作业中引用的文件名匹配；`New_Employees.CSV` 也是潜在匹配项，因为它的名称与 *_DIMENSIONNAME*.CSV 模式匹配。

示例

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

importOwnershipData

将环境中可用的 CSV 文件中的所有权数据导入到期间。

运行此命令之前，服务管理员使用 uploadFile 命令将导入源 CSV 文件加载到环境中。此 CSV 文件的标题如下：

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

`POwn`、`Control` 和 `Method` 为可选值。

导入的所有权数据会与任何现有数据合并，这可能会产生无效的所有权条目。如果一个实体存在于一个层次的多个分支中，则导入的所有权数据可能导致实体的合并所有权百分比超过 100%。您必须手动更正所有权百分比，以确保其不超过 100%。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 对实体具有写入访问权限的用户。

用法

`epmautomate importOwnershipData *Scenario Year Period FILE_NAME*`，其中：

- `Scenario` 是要将所有权数据导入到的方案。
- `Year` 是要将数据导入到的年份。
- `Period` 是要将所有权数据导入到的年份的期间。
- `FILE_NAME` 是要从中导入数据的 CSV 文件的名称。

示例

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

importPreMappedBalances

从 Account Reconciliation 存储库的文件中导入预映射的余额数据。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate importPreMappedBalances *PERIOD FILE_NAME BALANCE_TYPE CURRENCY_BUCKET*`，其中：

- *`PERIOD`* 是期间名称
- *`FILE_NAME`* 是包含要导入数据的 CSV 文件的名称
- *`BALANCE_TYPE`* 是 `SRC` 或 `SUB`
- *`CURRENCY_BUCKET`* 是 `Entered`、`Functional` 或 `Reporting`

示例

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

importPreMappedTransactions

从 Account Reconciliation 存储库中的 CSV 文件导入预映射的事务。

适用于

Account Reconciliation

所需角色

服务管理员、超级用户、用户、查看者
具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate importPreMappedTransactions` *`PERIOD TRANSACTION_TYPE FILE_NAME DATE_FORMAT`*，其中：

- *`PERIOD`* 是期间名称
- *`TRANSACTION_TYPE`* 是下列值之一：
  - `BEX` 用于加载余额解释
  - `SRC` 用于加载源系统调整
  - `SUB` 用于加载子系统调整
  - `VEX` 用于加载差异分析解释
- *`FILE_NAME`* 是要从中导入数据的 CSV 文件的名称
- *`DATE_FORMAT`* 是日期格式文本字符串；例如，MMM d, yyyy。

示例

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

importProfiles

从 Account Reconciliation 存储库中的 CSV 文件导入新的配置文件定义。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate importProfiles` *`FILE_NAME PROFILE_TYPE METHOD DATE_FORMAT`*，其中：

- *`FILE_NAME`* 是要从中导入数据的 CSV 文件的名称
- *`PROFILE_TYPE`* 是 `profiles` 或 `children`
- *`METHOD`* 是 `Replace` 或 `Update`
- *`DATE_FORMAT`* 是日期格式文本字符串；例如，MMM d, yyyy

示例

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

importRates

从 Account Reconciliation 存储库中的 CSV 文件导入汇率。

适用于

Account Reconciliation

所需角色

服务管理员、超级用户、用户、查看者
具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate importRates` *`PERIOD RATE_TYPE REPLACE_MODE FILE_NAME`*，其中：

- *`PERIOD`* 是期间名称
- *`RATE_TYPE`* 是预定义的汇率类型
- *`REPLACEMENT_MODE`* 是 `Replace` 或 `ReplaceAll`
- *`FILE_NAME`* 是要从中导入汇率的 CSV 文件的名称

示例

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

importRCAttributeValues

将属性值导入到 Account Reconciliation“调节合规性”列表或组属性中。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户（需要通过 ACL 提供的额外安全性）

用法

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

- *`ATTRIBUTE_NAME`* 是要向其导入值的列表或组属性的名称。
- *`FILE_NAME`* 是要从其导入值的 CSV 导入文件。在运行此命令之前，先使用 uploadFile 命令将此文件上传到环境。
- `METHOD`（可选）是值的导入方式。有效值：
  - `Replace`：将导入文件中的所有值作为属性值添加到“调节合规性”中。现有属性值将替换为导入文件中的值；属性中不存在但导入文件中存在的值将添加进来。属性中存在但导

入文件中不存在的值将保持不变。请注意，特定键值的所有属性数据都将替换为文件中的内容或被清除。新值将按其在文件中的顺序添加在底部。
当您仅从源系统移动最新更改时（例如，从某个购置项添加新存储数据以仅将指定的属性值（如果存在）替换为导入文件中的值），这种类型的导入非常有用。这是默认设置。

  - `Replace All`：将现有属性值替换为导入中的值。属性中存在但导入文件中不存在的值将被删除。
    要通过完整更新从源系统镜像值时（例如，要完成每周更新以与源系统中的存储数据同步），这种类型的导入非常有用。
  - `Update`：用导入文件中的所有值替换属性值或将其添加到属性。现有属性值将替换为导入文件中的值。导入文件中存在但属性中不存在的值将添加进来。属性中存在但导入文件中不存在的值将保持不变。仅特定键值的属性数据将替换为文件中的内容；文件中没有的属性的数据将保持不变。导入文件中存在但属性中不存在的任何键都会导致出错。
    要更新所有属性值中的一些属性时（例如，在重新组织后更新存储管理器但不影响其余存储数据时），这种类型的导入非常有用。

- `Dateformat`（可选）指定要解析的有效日期格式（例如 `DD/MM/YYYY`、`DD-MMM-YYYY`（默认值）、`MMM d,yyyy` 和 `All`）。可以指定用分号分隔的多个日期格式值。

示例

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

importReconciliationAttributes

使用 uploadFile 命令将调节属性从服务管理员上传到 Account Reconciliation 环境的文件导入现有调节中。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

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

- `FILE` 是包含要导入调节的调节属性的 CSV 文件的名称。
- `Period` 标识调节所属的期间。
- `Rules`（可选）标识在导入属性之后，要对受影响的调节运行的规则。使用逗号分隔多个规则名称。有效值为：
  - `None`：不对受影响的调节运行任何规则。这是默认值；不得与其他值结合使用。

  - `ALL`：针对指定期间，运行为调节定义的所有规则。此值必须单独使用，不能与其他规则名称结合使用。
  - `SET_ATTR_VAL`：运行预定义的规则以设置属性值。
  - `CRT_ALT`：运行预定义的规则以创建警报。
  - `AUTO_APP`：运行预定义的规则以自动批准调节。
  - `AUTO_SUB`：运行预定义的规则以自动提交调节。
  - `EMAIL_ON_SAVE`：运行预定义的规则以在更新调节之后自动发送电子邮件。
- `Reopen`（可选）指定在完成导入操作时是否重新打开已更改的调节。默认值为 `false`。
- `Dateformat`（可选）指定要解析的有效日期格式（例如，`MM-dd-yyyy`、`dd-MMMM-yy`、`MMM d` 和 `yyyy`）。可以指定用分号分隔的多个日期格式值。

示例

- **导入期间的属性值并运行具有多种日期格式的多个规则：**
  ```
  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"
  ```
- **导入期间的属性值而不运行规则：**
  ```
  epmAutomate importReconciliationAttributes Reconciliations.csv "July 2020"
  ```
- **导入期间的属性值，运行所有适用的规则和重新打开受影响的调节：**
  ```
  epmAutomate importReconciliationAttributes Reconciliations.csv "July 2020"
  Rules=ALL Reopen=true
  ```

importSnapshot

将快照的内容导入服务环境中。导入的快照必须位于默认上传位置中。

服务管理员使用 uploadFile 命令上传快照或使用 copySnapshotFromInstance 命令从其他实例复制该快照。

- 以下内容不属于 Planning、Planning 模块和自由形式应用程序快照的一部分：
  - 审核数据
  - 作业控制台数据

  如果要将审核和作业控制台数据复制到目标环境，请使用 cloneEnvironment 命令或克隆环境功能。
  如果 Planning 业务流程包含重命名的植入期间成员而原始成员已被自定义期间成员取代，快照导入可能会失败。例如，您将植入的 *YearTotal* 期间成员重命名为 *unused_YearTotal* 后，添加了使用原始植入成员名称（在此示例中为 *YearTotal*）的备用类型期间成员。在这种情况下，快照导入可能会失败。
- 快照不包含数据管理临时表数据。要导入此数据，请使用 exportDataManagement 和 importDataManagement 命令或数据管理系统维护脚本界面。您可以使用 cloneEnvironment 命令或克隆环境功能来创建环境的相同副本，包括数据管理临时表数据。

您使用此命令可以完成的活动取决于您的角色。

- 服务管理员只能将应用程序对象导入环境。
- 您需要同时具有服务管理员和身份域管理员角色才能将应用程序内容导入到服务环境，并将身份域对象（用户及其预定义角色分配）导入到环境的身份域。

如果正在导入的快照中引用了不在身份域中的用户，则此命令将在身份域中创建一个用户，并为每个用户分配您在命令中指定的默认密码，如果您没有在命令中指定密码，则会为每个用户分配一个临时的唯一密码。默认情况下，用户必须在首次登录期间重置密码。

> **注：**
>
> - 对于除 Account Reconciliation、Profitability and Cost Management 和 Oracle Fusion Cloud Enterprise Data Management 以外的业务流程：在加载元数据时，如果前一次尝试由于共享成员在大纲中位于基本成员之前而导致记录被拒绝，则 Oracle Fusion Cloud Enterprise Performance Management 可能会进行多次加载传递。这类尝试可能会增加命令处理时间。
> - 如果用户属于访问控制中的组的成员，则必须分配有预定义角色。不允许尝试将未分配预定义角色的用户分配到组。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Enterprise Data Management Cloud、销售规划 和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

身份域管理员角色是导入用户和预定义角色分配所必需的。

用法

`epmautomate importSnapshot` *`SNAPSHOT_NAME`* `[importUsers=true|false] [userPassword=`*`DEFAULT_PASSWORD`*`] [resetPassword=true|false]`，其中：

- *`SNAPSHOT_NAME`* 是位于默认上传位置的快照的名称。
- `importUsers`（可选）指定是否从快照中导入用户及其预定义的角色分配。默认值为 `false`。如果源快照包含有关新用户的数据，或者已将新角色分配给当前用户，则使用 `importUsers=true` 将用户和预定义角色分配导入到身份域。
  用户登录值不区分大小写。例如，用户登录值 `jane.doe@example.com` 被视为与 `Jane.Doe@Example.com` 或其任何大小写变体相同。如果任何变体与身份域中现有的用户登录匹配，则此命令不会从快照导入该用户。

> **注：**
>
> - 如果执行导入操作的用户不是身份域管理员，则导入用户及其预定义角色将失败。迁移状态报表中将记录以下错误：无法导入外部目录对象 ARTIFACT_NAME。用户 USER_NAME 无权执行此操作。用户必须具有身份域管理员角色才能执行此操作。
> - 如果您不导入用户，并且源快照中的用户未分配目标环境中的预定义角色，则将显示错误 (EPMIE-00070: Failed to find user during assigned roles import)。

  - 对用户预定义角色的更改将根据源快照中分配的角色进行更新。但是，不会为了匹配源快照中的角色分配而删除目标中的角色分配。例如，假设 jdoe 分配给目标环境中的超级用户预定义角色，但在源快照中只有用户角色。在这种情况下，此命令会将 jdoe 分配给用户角色，并且不会删除目标环境中的超级用户角色分配。
  - 如果源快照中不存在现有用户，此命令不会从目标环境中删除这些用户。例如，jdoe 在目标环境中有一个帐户，但此帐户在源快照中不存在。在这种情况下，目标环境中 jdoe 的该帐户不会被删除。
  - 此命令添加目标环境中不存在的用户；它不会更新目标环境中的当前用户属性，即使源快照中的这些属性不同。例如，如果源快照中 jdoe 的姓氏在目标环境中拼写不同，则不会在目标环境中进行更改。
  - 此命令不会更改目标环境中现有用户的密码，即使它在源快照中有所不同。
- userPassword（可选）指示要分配给在身份域中创建的新用户的默认密码。您指定的密码必须符合密码的最低要求。如果没有为该参数指定值，则会为每个用户分配一个唯一的临时密码。
- *resetPassword*（可选）指示新用户是否必须在首次登录时更改密码。默认值为 true，要求新用户在首次登录时更改密码。如果此值设置为 true，则新用户将收到提示其更改密码的帐户激活电子邮件。

示例

- 仅导入应用程序对象：`epmautomate importSnapshot April16FullApp`
- 导入应用程序和身份域对象（需要服务管理员和身份域管理员角色）：
  - 为每个新用户分配一个唯一的临时密码，并强制新用户在首次登录之后重置其密码：
    ```
    epmautomate importSnapshot April16FullApp importUsers=true
    ```
  - 分配特定的密码并允许用户不更改密码（如果他们选择这样做）。对于向生产环境的导入，不建议使用此方式：
    ```
    epmautomate importSnapshot April16FullApp importUsers=true
    userPassword=P@ssw0rd1 resetPassword=false
    ```

importSupplementalCollectionData

将补充集合数据从文件导入应用程序。

使用 uploadFile 命令将包含数据的文件上传到默认上传位置。导入文件的格式如下所示：

```
#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
```

例如：

```
#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
```

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

服务管理员

用法

> **注：**
>
> 所有命令参数都必须用双引号括起来。

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

- `FILE_NAME` 是 CSV 文件的名称，该文件位于默认上传位置，其中包含正确设置了格式的补充数据。
- `COLLECTION_NAME` 是应将文件中的补充数据导入到的集合的名称。
- `YEAR` 是要用于集合的年份维成员。
- `PERIOD` 是要用于集合的期间维的名称。
- *`FREQUENCY_DIMENSION`*（可选）是要用于集合的频率维的名称。可以按 *`"FREQUENCY_DIMENSION1=MEMBER" "FREQUENCY_DIMENSION2=MEMBER"`* 格式指定所需的任意数量的频率维。

示例

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

importSupplementalData

将补充数据从文件导入应用程序。

使用 uploadFile 命令将包含数据的文件上传到默认上传位置。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

服务管理员

用法

> **注：**
>
> 所有命令参数都必须用双引号括起来。

`epmautomate importSupplementalData "FILE_NAME" "DATA_SET_NAME" "YEAR" "PERIOD_NAME" "SCENARIO_NAME"`，其中：

- `FILE_NAME` 是 CSV 文件的名称，该文件位于默认上传位置，其中包含正确设置了格式的补充数据。
- `DATA_SET_NAME` 是文件中的补充数据应导入的数据集的名称。
- `YEAR` 是部署的数据集所对应的年份。
- `PERIOD_NAME` 是部署的数据集所对应的期间的名称。
- `SCENARIO_NAME` 是部署的数据集所对应的方案的名称。

示例

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

importTemplate

通过导入 profitinbox 中存在的模板文件来创建应用程序结构。

服务管理员使用 uploadFile 命令将模板文件上传到 profitinbox 中。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate importTemplate APPLICATION_NAME File_Name isApplicationOverwrite=true|false`，其中：

- *APPLICATION_NAME* 是要通过导入模板创建的 Profitability and Cost Management 应用程序的名称
- `File_Name` 是包含应用程序模板的 .ZIP 文件的名称。此文件必须存在于 profitinbox 中。

- `isApplicationOverwrite` 指定是否覆盖现有应用程序（如果有）。请以全部小写形式指定此参数值。

示例

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

importTMAttributeValues

将值导入到 Account Reconciliation“事务匹配”组属性中。

运行此命令之前，服务管理员使用 uploadFile 命令将包含事务匹配属性值的导入文件上传到环境。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户（需要通过 ACL 提供的额外安全性）

用法

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

- *`ATTRIBUTE_NAME`* 是要向其导入值的组属性的名称。
- *`FILE_NAME`* 是要从其将值导入到“事务匹配”中的 CSV 导入文件。
- `METHOD`（可选）是值的导入方式。有效值：
  - `Replace`：将导入文件中的所有值添加到“事务匹配”组属性中。现有属性值将替换为导入文件中的值；属性中不存在但导入文件中存在的值将添加进来。属性中存在但导入文件中不存在的值将保持不变。请注意，特定键值的所有属性数据都将替换为文件中的内容或被清除。新值将按其在文件中的顺序添加在底部。
    当您仅从源系统移动最新更改时（例如，从某个购置项添加新存储数据以仅将指定的属性值（如果存在）替换为导入文件中的值），这种类型的导入非常有用。这是默认设置。
  - `Replace All`：将现有属性值替换为导入中的值。属性中存在但导入文件中不存在的值将被删除。
    要通过完整更新从源系统镜像值时（例如，要完成每周更新以与源系统中的存储数据同步），这种类型的导入非常有用。
  - `Update`：用导入文件中的所有值替换属性值或将其添加到属性。现有属性值将替换为导入文件中的值。导入文件中存在但属性中不存在的值将添加进来。属性中存在但导入文件中不存在的值将保持不变。仅特定键值的属性数据将替换为文件中的内容；文件中没有的属性的数据将保持不变。导入文件中存在但属性中不存在的任何键都会导致出错。
    要更新所有属性值中的一些属性时（例如，在重新组织后更新存储管理器但不影响其余存储数据时），这种类型的导入非常有用。
- `Dateformat`（可选）指定要解析的有效日期格式（例如 `DD/MM/YYYY`、`DD-MMM-YYYY`（默认值）、`MMM d,yyyy` 和 `All`）。可以指定用分号分隔的多个日期格式值。

示例

```
epmautomate importTMAttributeValues TMGA TMGA.csv METHOD=Replace DATEFORMAT="All"
```

importTmPremappedTransactions

针对特定的数据源，将预映射的事务数据从 Account Reconciliation 存储库中的文件导入事务匹配。

服务管理员使用 uploadFile 命令将事务文件上传到服务。

此命令会在控制台中显示导入状态和导入日志文件名。服务管理员使用 downloadFile 命令将日志文件下载到本地计算机。

有关导入文件格式要求以及导入数据的相关信息，请参阅《使用 *Account Reconciliation* 调节帐户》中的“导入数据”。

> **注：**
>
> - 一次只能为一个匹配类型导入事务。但是，并行导入可能会遇到不同的匹配类型。
> - 与在“作业”屏幕上不同，您一次只能从一个文件中导入预映射的事务数据。
> - 为所有数据源导入预映射的事务后，运行 `runautomatch` 命令。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate importTmPremappedTransactions` *`MATCH_TYPE DATA_SOURCE FILE_NAME [DATE_FORMAT]`*，其中：

- `MATCH_TYPE` 是在 Account Reconciliation 中定义的匹配类型。
- `DATA_SOURCE` 是与您指定的调节类型关联的数据源的标识符。
- `FILE_NAME` 是 CSV 文件的名称，该文件中包含要导入的事务。该文件必须在服务中。
- `DATE_FORMAT` 是一个可选的参数，指示事务导入文件中包含的日期字段的格式。默认为 `dd-MMM-YYYY`。其他受支持的日期格式为：`MM/dd/yyyy`、`dd/MM/yyyy`、`MM-dd-yyyy`、`d-M-yyyy` 和 `MMM d.yyyy`。

示例

```
epmautomate importTmPremappedTransactions "INTERCOMPANY" "AP" dailydata.csv d-M-yyyy
```

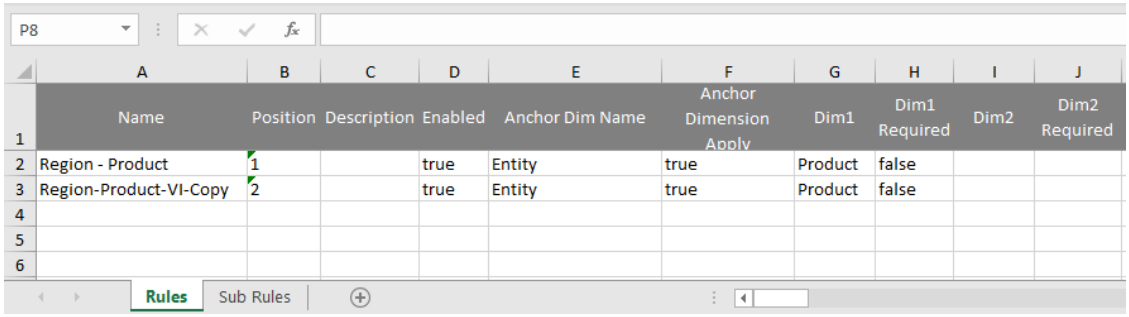
importValidIntersections

将有效交叉点组从 ZIP 文件导入业务流程中，该 ZIP 文件含有一个包括有效交叉点定义的 Excel 文件。在运行此命令之前，请先使用 uploadFile 命令将导入文件上传到环境。

您的导入 ZIP 文件应当包含一个 Excel 文件，该 Excel 文件应具有两个工作表（Rules 工作表和 Sub Rules 工作表）才能成功导入有效交叉点。第一个表（即 Rules 表）应当定义交叉点组、所包括的维、属性（如“未指定的维有效”和“需要其他维”）。第二个表（即 Sub Rules 表）应当提供成员选择和排除。有关更多信息，请参阅《管理 *Planning*》中的下列主题。

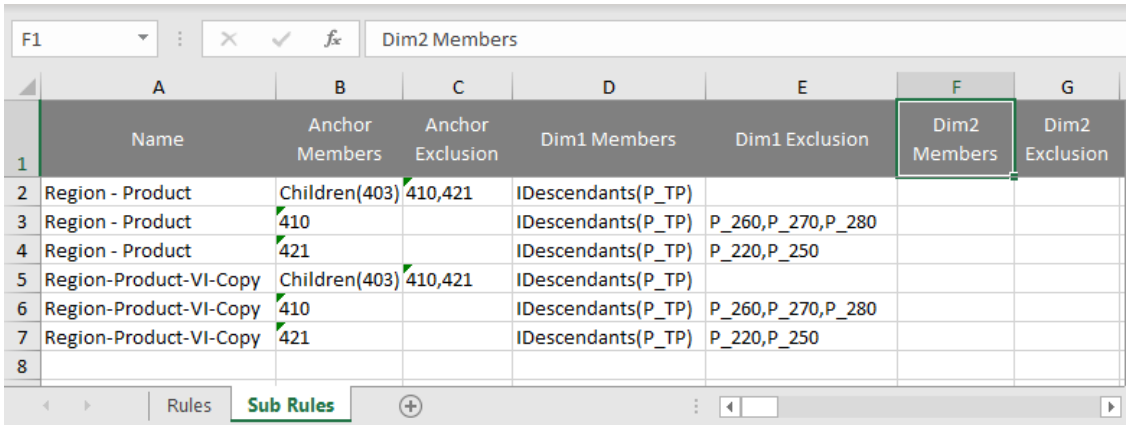
- 锚点和非锚点维
- 有效交叉点示例

获取导入文件格式模板的最佳方式是从应用程序中导出有效交叉点。下图中显示了示例格式。



	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

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate importValidIntersections` *`FILE_NAME`*`.zip [ErrorFile=`*`ERROR_FILE_NAME`*`.txt]`，其中：

- FILE_NAME 是 ZIP 文件的名称，该 ZIP 文件含有一个包括有效交叉点定义的 Excel 文件。
- ErrorFile（可选）标识要向其中写入错误记录的文本文件的名称。如果未指定此参数值，该命令会自动生成一个错误文件；您可以在作业控制台中查看该错误文件的名称。

示例

```
epmautomate importValidIntersections VI_Import_File.zip
ErrorFile=VI_Import_Log.txt
```

invalidLoginReport

在 OCI（第 2 代）环境中创建无效登录报表，该报表列出在指定的一段时间（对应为环境指定的审核保留期）内失败的环境登录尝试。默认保留期为 30 天。最多可以将其延长到 90 天，方法是在 Oracle Cloud Identity 控制台中更改**审计保留期（天）**设置。要将审核数据保留 90 天以上，请定期下载并存档此报表和角色分配审核报表。

无效登录报表包含如下信息：

- 尝试登录的用户的用户名
- 用户尝试从其登录的远程 IP 地址
- 登录尝试的时间戳

此报表显示登录相应身份云服务失败的所有尝试。这些尝试可能并非都与一个 Oracle Fusion Cloud Enterprise Performance Management 实例相关。

	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

使用 downloadFile 命令将该报表下载到本地计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

身份域管理员和任何预定义角色

用法

`epmAutomate invalidLoginReport` *FROM_DATE TO_DATE FILE_NAME*`.CSV`，其中：

- *FROM_DATE* 指明要生成报表的期间的开始日期（格式为 YYYY-MM-DD）。此数据必须位于 Oracle Cloud Identity Console 中指定的审核数据保留期间内。
- *TO_DATE* 指明要生成报表的期间的结束日期（格式为 YYYY-MM-DD）。
- *FILE_NAME* 是报表的 CSV 文件名。

> **Note:**
>
> 只能为过去 90 天生成此报表。

示例

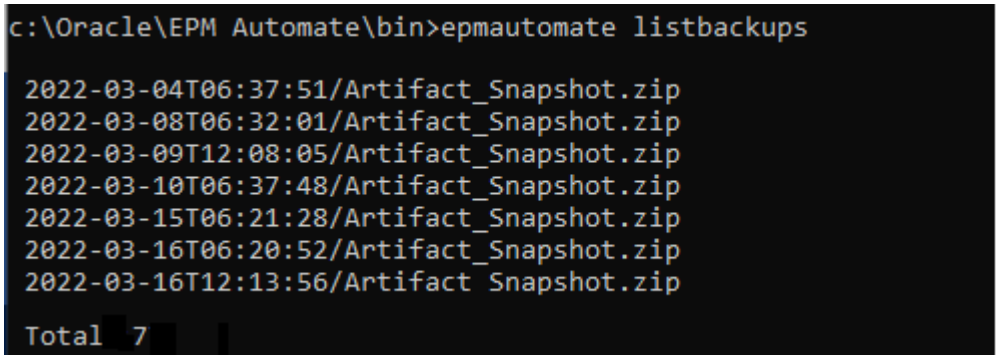
```
epmAutomate invalidLoginReport 2021-06-01 2021-06-30 invalidLoginReport.CSV
```

listBackups

列出环境的可用备份快照，以确定特定备份是否可用，以便您可以对其进行存档或使用它来自行还原当前环境。

尝试还原特定备份之前，使用此命令检查 Oracle Object Storage 中是否有所需的备份。如果备份可用，您可以通过运行 restoreBackup 命令来还原它（将其复制到您的环境中）。复制备份后，可以使用 importSnapshot 命令将其导入。环境的自助还原可以节省您的处理时间。

对于除 Narrative Reporting 以外的服务，此命令使用命名约定 `YYYY-MM-DDTHH:MM:SS/Artifact_Snapshot.zip` 列出可用的备份快照（由日常维护过程创建）；例如，`2022-02-16T21:00:02/Artifact_Snapshot.zip`。对于 Narrative Reporting，可用快照使用命名约定 `YYYY-MM-DDTHH:MM:SS/EPRCS_Backup.tar.gz`；例如，`2022-02-16T21:00:02/EPRCS_Backup.tar.gz`。在这两种情况下，时间戳都会反映创建快照时的 UTC 时间。下图显示了示例命令输出。



适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

```
epmAutomate listBackups
```

示例

```
epmAutomate listBackups
```

listFiles

列出默认位置、数据管理文件夹和 profitinbox/profitoutbox (Profitability and Cost Management) 中的文件的名称。

此命令同时也列出增量和备份导出文件、迁移快照、访问日志和活动报表。此图显示了命令输出的截断版本。

```
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
```

如果在生成环境快照时执行此命令，则此命令不会列出当前快照；例如，在日常维护期间。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

```
epmautomate listFiles
```

示例

```
epmautomate listFiles
```

loadData

使用 profitinbox 中的文件将数据加载到计算多维数据集中。

服务管理员使用 uploadFile 命令将文件加载到 profitinbox 中。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate loadData` *`APPLICATION_NAME`* `dataFileName=`*`File_Name`* `PARAMETER=`*`VALUE`*，其中：

- *`APPLICATION_NAME`* 是要将数据加载到其中的 Profitability and Cost Management 应用程序的名称
- `dataFileName=`*`File_Name`* 指定 profitinbox 中的可用数据加载文件。
  数据文件名必须用双引号括起来。
- `PARAMETER=`*`VALUE`* 指明用于加载数据的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：
  - `clearAllDataFlag=true|false` 指定是否清除应用程序多维数据集中的现有数据
  - `dataLoadValue=OVERWRITE_EXISTING_VALUES|ADD_TO_EXISTING` 指定如何处理现有数据

示例

```
epmautomate loadData BksML12 dataFileName="data1.txt"clearAllDataFlag=true
dataLoadValue="OVERWRITE_EXISTING_VALUES"
```

loadDimensionViewpoint

使用定义的加载将数据从加载文件加载到一个未绑定、已绑定或部分绑定的视点（一部分节点）中。

必须在加载视点的环境中提供加载文件、CSV 文件、Excel (XLSX) 文件或包含一个 CSV 或 XLSX 文件的 ZIP 文件。可以使用 uploadFile 或 copyFileFromInstance 命令将加载文件上传到环境中。

要使用默认加载定义加载视点，请使用 loadViewpoint 命令。

适用于

Oracle Fusion Cloud Enterprise Data Management。

所需角色

服务管理员

用法

`epmautomate loadDimensionViewpoint APPLICATION_NAME DIMENSION_NAME FILE_NAME LOAD_NAME [loadOption=ReplaceNodes|Merge] [purpose="PURPOSE"]`，其中：

- `APPLICATION_NAME` 是 Oracle Enterprise Data Management Cloud 应用程序的名称。
- `DIMENSION_NAME` 是要加载的维的名称。
- `FILE_NAME` 是要从中加载视点的文件的名称（包括扩展名：CSV、XLSX 或 ZIP）。
- `LOAD_NAME` 是用于加载视点的已定义加载的名称。
- `loadOption`（可选）标识如何加载视点。有效的加载选项如下：
  - `ReplaceNodes`，可从层次中清除加载文件中关系以外的所有其他关系（包括孤立关系和具有相同层次集的其他视点使用的关系）。这是默认加载类型。
  - `Merge`，可以通过仅处理增量更改来保留现有关系。
- `purpose`（可选）是一个用双引号括起来的文本字符串，用于标识加载该视点的原因。

示例

- 加载视点以替换现有层次：`epmautomate loadDimensionViewPoint "Data Warehouse" Citizen Data_Warehouse_Citizen_20241108.csv DW-Citizen`
- 通过将增量更改与指定加载合并来加载视点：`epmautomate loadDimensionViewPoint"Data Warehouse" Citizen Data_Warehouse_Citizen_20241108.csv DW-Citizen loadOption=Merge purpose="Load Test"`

loadDimData

将维元数据从 profitinbox 中的一个或多个文件加载到应用程序中。

服务管理员使用 uploadFile 命令将元数据文件加载到 profitinbox 中。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate loadDimData APPLICATION_NAME dataFileName=File_Name [stringDelimiter="DELIMITER"] [acceptableDecreasePercentage=PERCENTAGE]`，其中：

- `APPLICATION_NAME` 是要将维元数据加载到其中的 Profitability and Cost Management 应用程序的名称
- `dataFileName` 指定 profitinbox 中可用的维元数据加载文件。要从多个文件加载元数据，请列出各个文件名，并用分隔符来分隔
- `stringDelimiter`（可选）指定用于分隔元数据文件名的分隔符。分隔符必须用双引号括起来。

- `acceptableDecreasePercentage`（可选）指定操作可接受的成员计数差异百分比（不带 % 符号）。如果传入文件中的新成员计数小于现有成员计数，则此值表示可接受的百分比减少。如果成员计数偏差超过此百分比，维数据加载将失败。

示例

```
epmautomate loadDimData BksML12 dataFileName="dimdata1.txt#dimdata1.txt"
stringDelimiter="#" acceptableDecreasePercentage=5
```

loadViewpoint

将加载文件中的视点（一组节点）加载到 Oracle Fusion Cloud Enterprise Data Management 应用程序中。

使用视点加载，您可以将数据加载到未绑定、已绑定或部分绑定的视点中。必须在加载视点的环境中提供视点加载文件、CSV 文件、Excel (XLSX) 文件或包含一个 CSV 或 XLSX 文件的 ZIP 文件。可以使用 uploadFile 或 copyFileFromInstance 命令将加载文件上传到环境中。

适用于

Oracle Enterprise Data Management Cloud

所需角色

服务管理员

用法

```
epmautomate loadViewpoint VIEW VIEWPOINT PURPOSE FILE_NAME
[loadType=ReplaceNodes|Merge],
```

其中：

- `VIEW` 是 Oracle Enterprise Data Management Cloud 视图的名称。
- `VIEWPOINT` 是要加载的视点的名称。
- `PURPOSE` 是用双引号括起来的文本字符串，用于说明加载视点的原因。
- `FILE_NAME` 是要从中加载视点的文件的名称，带有扩展名。
- `loadType`（可选）标识如何加载视点。有效值包括 `Merge` 和 `ReplaceNodes`。
  - 使用 `Merge` 可以通过处理增量更改来保留现有关系。
  - 使用 `ReplaceNodes` 将从层次中清除加载文件中关系以外的所有其他关系（包括孤立关系和具有相同层次集的其他视点使用的关系）。这是默认加载类型。

示例

- **合并增量更改：** `epmautomate loadViewpoint USOperations Entity "Daily Upstream Load" data_Entity.CSV loadType=Merge`
- **替换现有层次：** `epmautomate loadViewpoint USOperations Entity "Replace US Operations data" data_Entity.CSV`

login

建立与环境的安全连接。此命令支持使用以下方式登录到环境：明文密码，或者包含密码或 OAuth 2.0 刷新令牌的加密密码文件。仅 OCI（第 2 代）环境支持使用 OAuth 2.0 刷新令牌登录。

需要登录来启动会话，会话在您注销之前将保持活动状态。

> **注：**
>
> - 对于已设置为使用多因素身份验证 (MFA) 进行基本身份验证的用户，不支持此命令。
> - EPM Automate 不支持使用组织的 SSO 凭据登录。
> - EPM Automate 不能与 SOCKS 代理一起使用，只能与 HTTP/HTTPS 代理一起使用。
> - 在批处理文件中使用此命令自动执行活动时，Oracle 建议您使用加密的密码或 OAuth 2.0 刷新令牌以避免在批处理文件中记录明文密码。
> - 在 Windows 计算机上，此命令会自动找到缺失的可能会阻止您建立连接的代理服务器中间安全证书，并将其添加到安装在 `C:\Oracle\EPM Automate` 下的 JRE。这可以防止在使用代理服务器访问 Internet 时出现与安全证书相关的登录错误。在 Linux 计算机上，login 命令从代理服务器找到缺失的安全证书，下载该证书，并显示错误。然后，具有 `root` 访问权限的用户可以将下载的证书安装在环境变量中标识的 `JAVA_HOME` 中可用的 JRE 中。请参阅以下信息源：
>   - Java 运行时环境和 EPM Automate
>   - Keytool Java 文档

如果您使用的是较旧版本，登录时会显示升级 EPM Automate 的消息。可以使用 upgrade 命令对安装进行静默升级。

如果您计划运行 addUsers、removeUsers、assignRole 或 unassignRole 命令，请不要使用 OAuth 刷新令牌登录。这些命令要求您使用基本身份验证。所有其他命令都适用于 OCI（第 2 代）环境中的 OAuth 2.0。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

- **使用未加密的密码：** `epmautomate login USERNAME PASSWORD URL [IDENTITYDOMAIN] [ProxyServerUserName=PROXY_USERNAME ProxyServerPassword=PROXY_PASSWORD ProxyServerDomain=PROXY_DOMAIN] ] [KeystorePassword=PASSWORD]`
- **使用加密的文件：** `epmautomate login USERNAME PASSWORD_FILE URL [IDENTITYDOMAIN] [ProxyServerUserName=PROXY_USERNAME] [ProxyServerPassword=PROXY_PASSWORD] [ProxyServerDomain=PROXY_DOMAIN] [KeystorePassword=KEYSTORE_PASSWORD]`

在这些命令中：

- *USERNAME* 是用户的用户名。
- *PASSWORD* 是用户的密码。
- *PASSWORD_FILE* 是用于存储用户的加密密码或 OAuth 2.0 刷新令牌的文件的名称和位置。请参阅 encrypt 命令。
- *URL* 是要连接的环境的基本 URL。您可以使用自定义或虚名 URL 来代替 Oracle Fusion Cloud Enterprise Performance Management URL。请参阅《管理员入门指南》中的“使用虚名 URL”

> **注：**
>
> 如果使用 API 网关或反向代理，则使用其 URL 和为您的环境定义的上下文来代替云 EPM URL。

- *IDENTITYDOMAIN*（可选）是环境的身份域。
  此值自动派生自云 EPM URL；将忽略您指定的任何值。
- `ProxyServerUserName` 是用户名，用于在控制 Internet 访问的 HTTP 代理服务器中对安全会话进行身份验证。在不添加域名前缀的情况下指定用户名。仅当在代理服务器上对您的网络启用了身份验证时才需要。
- `ProxyServerPassword` 是用于在代理服务器中对用户进行身份验证的密码。仅当在代理服务器上对您的网络启用了身份验证时才需要。可对此密码进行加密。请参阅 encrypt 命令。如果此密码已加密，则从 *PASSWORD_FILE* 读取。
- `ProxyServerDomain` 是为 HTTP 代理服务器定义的域的名称（而非服务器名称或代理服务器主机名）。仅当在代理服务器上对您的网络启用了身份验证并且配置了代理服务器域时才需要。
- `KeystorePassword`（可选）是导入代理服务器安全证书所需的密钥库密码。仅在 Windows 上使用此参数，并且仅当在使用代理服务器来引导 Internet 访问的环境中遇到以下错误时使用此参数：

  `EPMAT-7: 由于密钥库中缺少一些 SSL 证书而无法连接`

  `EPMAT-7: 由于密钥库中缺少上述 SSL 证书而无法连接`

> **注：**
>
> EPM Automate 检测并使用您计算机上的 HTTP/HTTPS 代理设置。
> EPM Automate 支持使用以下身份验证机制连接到代理服务器：
>
> - 基本身份验证
> - 摘要身份验证
> - Kerberos 身份验证
> - 协商代理身份验证
> - NTLM 身份验证
>
> 可用的身份验证方法及其配置取决于您使用的代理服务器。
>
> 在 Linux 计算机上，如果代理设置要求您在代理服务器中进行身份验证，您必须输入代理服务器域、用户名和密码作为此命令的参数。如果您需要代理服务器域和凭据方面的帮助，请与您的网络管理员联系。

示例

- 使用未加密的云 EPM 密码，无代理身份验证：
  ```
  epmautomate login serviceAdmin P@ssword1 https://test-cloud-pln.pbcs.us1.oraclecloud.com
  ```
- 使用加密的文件，无代理身份验证：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com
  ```
- 在代理服务器上启用了身份验证且具有服务器域时，使用加密的文件：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com ProxyServerPassword=example ProxyServerDomain=example
  ```
- 在代理服务器上启用了身份验证但没有服务器域时，使用加密的文件：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com ProxyServerPassword=example
  ```
- 在代理服务器上启用了身份验证且具有服务器域时，使用加密的云 EPM 和代理服务器密码：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com ProxyServerDomain=example
  ```
- 在代理服务器上启用了身份验证但没有服务器域时，使用加密的云 EPM 和代理服务器密码：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://test-cloud-pln.pbcs.us1.oraclecloud.com ProxyServerUserName=john.doe@example.com
  ```
- 配合 APIGEE API 网关使用加密的文件：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://exampleapigee.apigee.com/epm example_ID_DOM
  ```
- 使用虚名 URL：
  ```
  epmautomate login serviceAdmin C:\mySecuredir\password.epw https://rebrand.ly/Automate
  ```

logout

终止当前与环境的连接。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

```
epmautomate logout
```

示例

```
epmautomate logout
```

maskData

遮蔽应用程序数据以确保数据私密性。仅在测试环境中使用此命令，以对应用程序开发人员隐藏敏感数据。

警告：请勿在生产环境中使用此命令，因为它会将当前的应用程序数据随机化，从而使其毫无意义。您无法撤消此命令的效果。如果您错误地遮蔽了服务环境中的数据，必须从备份或从维护快照还原数据。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate maskData [-f]`，其中 `-f` 是用于强制开始遮蔽过程而不经用户确认的选项。如果未使用 `-f` 选项，该命令将提示您确认操作。

示例

```
epmautomate maskData [-f]
```

mergeDataSlices

将聚合存储多维数据集的所有增量数据切片合并到主数据库切片中，（可选）并删除具有零值的单元格。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate mergeDataSlices CUBE_NAME [keepZeroCells=true|false]`，其中：

- `CUBE_NAME` 标识了所有数据切片将合并到的聚合存储多维数据集。
- `keepZeroCells`（可选）指定是否删除包含零值的单元格（逻辑清除区域中的数据将导致单元格的值为零）。默认值为 `true`

示例

```
epmautomate mergeDataSlices rep1 keepZeroCells=false
```

mergeSlices

将增量数据切片合并到主数据库多维数据集，并可以选择删除含 0（零）值的 Oracle Essbase 单元格，以精简多维数据集。

删除含 0 的单元格可优化多维数据集性能。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate mergeSlices applicationName [removeZeroCells=true|false]`，其中：

- `applicationName` 是 Profitability and Cost Management 应用程序的名称。
- `removeZeroCells` 为可选，用于指定是否删除含 0 的单元格。此参数的默认值为 `false`。

示例

- 合并切片但不删除含 0 的单元格：
  - `epmautomate mergeSlices BksML30`
  - `epmautomate mergeSlices BksML30 removeZeroCells=false`

- 合并切片并删除包含 0 的单元格：`epmautomate mergeSlices BksML30 removeZeroCells=true`

optimizeASOCube

优化查询性能，以选择从 ASO 多维数据集进行数据提取的聚合视图。

在因数据较大而导致默认聚合不足以满足数据提取或报告需求时，可使用此命令对 ASO 多维数据集执行查询优化操作。典型的优化过程如下：

- 删除默认聚合和基于查询的聚合。
- 启动查询跟踪。
- 从 Profitability and Cost Management 查询管理器、Oracle Smart View for Office 或数据管理运行示例查询，以及适合优化过程用于训练 Oracle Essbase 的查询类型的其他代表性 MDX 查询。
- 根据优化的查询或默认查询创建聚合。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate optimizeASOCube` *`APPLICATION_NAME OPTIMIZATION_TYPE`*，其中：

- *`APPLICATION_NAME`* 是 ASO 多维数据集所属的 Profitability and Cost Management 应用程序的名称。
- *`OPTIMIZATION_TYPE`* 是多维数据集优化操作。可接受的值包括：
  - `clearAggregations`，删除默认视图和基于查询的视图。
  - `createAggregations`，创建默认的 Essbase 聚合视图。使用此选项将执行默认聚合，而不是基于查询的聚合
  - `startQueryTracking`，启动查询跟踪。
  - `stopQueryTracking`，停止查询跟踪。使用此选项将使 Essbase 停止收集优化信息。Essbase 继续收集优化信息，直到停止查询跟踪或停止 Essbase。Essbase 会根据收集的数据聚合视图，直到停止查询跟踪。
  - `createQBOAggregations`，根据在启用查询跟踪后运行的已优化查询来创建 Essbase 聚合视图。

示例

- 删除默认聚合视图和基于查询的聚合视图：
  `epmautomate optimizeASOCube BksML12 clearAggregations`
- 启动查询跟踪
  `epmautomate optimizeASOCube BksML12 startQueryTracking`
- 根据在启动查询跟踪后运行的已优化查询来创建 Essbase 聚合视图：
  `epmautomate optimizeASOCube BksML12 createQBOAggregations`

programDocumentationReport

创建包含 Profitability and Cost Management 应用程序逻辑的程序文档报表。

可以使用 downloadFile 命令将报表下载到本地计算机。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

`epmautomate programDocumentationReport` *`APPLICATION_NAME POV_NAME`* `[fileName=`*`FILE_NAME`*`] [fileType=PDF|WORD|EXCEL|HTML] [useAlias=true|false] [skipFilters=true|false] stringDelimiter="`*`DELIMITER`*`"`，其中：

- *`APPLICATION_NAME`* 是要为其创建程序文档报表的 Profitability and Cost Management 应用程序的名称。
- *`POV_NAME`* 是要为其生成报表的应用程序中的模型 POV 名称。
- `fileName`（可选）是报表文件的唯一名称（包含扩展名）。默认报表文件名为 `HPCMMLProgramDocumentationReport_`*`APPLICATION_NAME_POV_NAME`*`.pdf`。
- `fileType`（可选）是输出文件格式。默认值为 `PDF`。
- `useAlias`（可选）指定是否打印别名来代替成员名称。默认值为 `false`。
- `skipFilters`（可选）指定是否忽略筛选器，为具有大量使用筛选器的规则的大型模型加快报表生成。默认值为 false。
  将此参数设置为 `true` 将跳过解析每个规则筛选器的过程，不会确定报表中估计的源计数和估计的目标计数的值。而是，该过程使用相应的 0 级成员计数。如果此参数值设置为 `false` 或未指定，则该命令将解析所有筛选器以生成更准确的计数。
- `stringDelimter` 是 POV 值中使用的分隔符。分隔符必须用双引号括起来。

示例

- 在解析规则筛选器后创建报表：
  ```
  epmautomate programDocumentationReport BksML30 2024_Feb_Actual fileName=Feb-Actual.xls fileType=Excel useAlias=true stringDelimiter="_"
  ```
- 创建报表以忽略筛选器：
  ```
  epmautomate programDocumentationReport BksML30 2024_Feb_Actual fileName=Feb-Actual.xls fileType=Excel useAlias=true skipFilters=true stringDelimiter="_"
  ```

provisionReport

生成角色分配报表（.CSV 文件）并将其存储在默认下载位置中。

该报表列出了预定义的角色（例如 Service-name 超级用户）和分配给用户的应用程序角色（例如 Planning 应用程序角色“批量分配”）。使用 downloadFile 命令下载报表。

可以生成两个版本的报表：简化版本和标准版本。简化报表与可从访问控制屏幕获取的角色分配报表相同，它不会列出纳入预定义角色的应用程序角色或分配给用户的应用程序角色的组件角色。此报表的标准版本将列出已纳入到可分配给用户的预定义角色中的组件角色。它还会列出分配给用户（直接或通过组）的应用程序角色。

生成此报表会刷新访问控制中提供的用户和角色信息。

仅适用于 OCI（第 2 代）：Oracle Fusion Cloud Enterprise Performance Management 将停用的用户视为未分配任何预定义角色的用户，即使此类用户在停用后可能具有预定义角色也是如此。此报表中不包括有关停用的用户的信息。

> **注：**
>
> 即将推出的版本中将弃用此命令，转而使用可以生成等效报表的 roleAssignmentReport 命令。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmautomate provisionReport` *`REPORT_NAME`* `[format=classic|simplified] [userType=serviceUsers|IDAdmins]`，其中：

- *`REPORT_NAME`* 是报表的名称。
- `format` 为可选，用于标识要对报表进行的格式设置。可接受的值：
  - `simplified` 是默认选项，将创建一个与从访问控制屏幕生成的角色分配报表相同的报表。
  - `classic` 将在创建的报表中列出已纳入到可分配给用户的预定义角色中的组件角色。它还会列出分配给用户（直接或通过组）的应用程序角色
- `userType` 为可选，用于标识要在报表中包含的用户。如果没有为此参数指定值，则使用默认值 `serviceUsers`。可接受的值：
  - `serviceUsers` 将在创建的报表中包含所有功能用户的相关信息（如果没有为身份域管理员分配一个预定义角色以授予应用程序访问权限，则不包含该类用户的相关信息）
  - `IDAdmins` 将在创建的报表中仅列出分配了身份域管理员角色的用户。此报表在采用标准格式和简化格式时完全相同

示例

- 创建标准报表：`epmautomate provisionReport myProvReport.CSV format=classic`
- 创建简化报表：
  - `epmautomate provisionReport myProvReport.CSV format=simplified`
  - `epmautomate provisionReport myProvReport.CSV userType=serviceUsers`
- 创建仅列出身份域管理员的报表：
  - `epmautomate provisionReport myProvReport.CSV userType=IDAdmins`
  - `epmautomate provisionReport myProvReport.CSV userType=IDAdmins format=classic`

purgeArchivedTmTransactions

从 Account Reconciliation 应用程序中清除存档的匹配事务。

使用 archiveTmTransactions 命令定期存档旧的匹配事务，然后运行此命令从 Account Reconciliation 中删除这些事务以确保应用程序保持最佳大小。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate purgeArchivedTMTransactions JobID=`*`JOB_ID`*，其中 `JobID` 是为存档匹配事务而运行的“存档 TM 事务”作业的标识符。当您执行 archiveTmTransactions 命令时，此作业 ID 显示在 EPM Automate 控制台中。您还可以在作业控制台中找到该 ID。

示例

```
epmautomate purgeArchivedTMTransactions JobID=100000002655003
```

purgeTmTransactions

从 Account Reconciliation 中删除匹配的事务。

适用于

Account Reconciliation

所需角色

- 服务管理员

- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate purgeTmTransactions matchType age [filterOperator=VALUE] [filterValue=VALUE] [logFilename=FILE_NAME]`，其中：

- `matchType` 是应删除匹配事务的匹配类型的标识符 (TextID)。
- `age` 标识自匹配事务以来的天数。将删除超过或等于此值的匹配事务。
- `filterOperator`（可选）是以下筛选条件之一，用于标识包含要删除的匹配事务的帐户。此值与 `filterValue` 相结合，用于标识应清除匹配事务的帐户：
  - `equals`
  - `not_equals`
  - `starts_with`
  - `ends_with`
  - `contains`
  - `not_contains`
- `filterValue`（可选）是用于标识要清除的事务的筛选器值。如果 `filterOperator` 为 `equals` 或 `not_equals`，则可以使用空格分隔的列表指定多个值；例如 `filterValue=101-120 filterValue=102-202`。如果指定多个值，将选择与任何筛选器运算符和筛选器值组合匹配的帐户中的事务进行清除。
- `logFilename`（可选）是用于记录命令活动相关信息的日志文件的名称。如果未指定文件名，将自动生成名为 `PurgeTransactions_JOB_ID` 的日志文件。

> **Note:**
>
> 如果未指定 `filterOperator` 和 `filterValue`，将从所有帐户中清除指定 `matchType` 的超过或等于 `age` 的所有匹配事务。

示例

- 为匹配类型 `cashrecon` 清除 `180` 天或更早的匹配事务：
  ```
  epmautomate purgeTMTransactions cashrecon 180 logFileName=tmlogs.log
  ```
- 为帐户 `101-120` 或 `102-202` 清除匹配类型 `cashrecon` 的 `180` 天或更早的匹配事务：
  ```
  epmautomate purgeTMTransactions cashrecon 180 filterOperator=equals
  filterValue=101-120 FilterValue=102-202
  ```
- 为包含字符串 `11` 的任何帐户清除匹配类型 `cashrecon` 的 `180` 天或更早的匹配事务：
  ```
  epmautomate purgeTMTransactions cashrecon 180 filterOperator=contains
  filterValue=11
  ```

recomputeOwnershipData

重新计算所有权数据

在以下情况下，需要重新计算 Financial Consolidation and Close 中的所有权数据：

- 添加或删除所有权管理帐户的覆盖规则之后
- 更改合并方法范围设置之后
- 数据库刷新之后（无论实体结构是否更改）

每次数据库刷新后，即使未更改实体结构，也需要重新计算 Tax Reporting 中的所有权数据。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 用户

用法

`epmautomate recomputeOwnershipData` *`Scenario Year Period`*，其中：

- `Scenario` 是要重新计算的方案的名称。
- `Year` 是要重新计算的年份。
- `Period` 是要重新计算的年份的第一个期间。
  将重新计算选定的期间和所有后续期间。

> **注：**
>
> 只有在重新计算所有权数据后，才能合并需要重新计算的 POV。

示例

```
epmautomate recomputeOwnershipData FCCS_total_Actual FY19 Jan
```

recreate

通过重新创建环境将环境还原到干净状态。

通过重新创建环境来完成以下任务：

- 在导入完整快照之前清理环境。
- 更改可在环境中部署的业务流程。

> **注意：**
>
> - 此命令将从环境中删除现有应用程序，以及所有用户定义的对象（可选）。此外，它还会重新创建数据库并删除所有现有数据。重新创建服务后，您可以使用迁移或 EPM Automate 创建新业务流程或导入业务流程。
> - 此命令会删除迁移历史记录。因此，迁移中提供的迁移状态报表将不包含历史信息。
> - 使用此命令之前，请先对环境执行完整备份。可执行 runDailyMaintenance 命令来创建备份快照。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate recreate [-f] [removeAll=true|false] [TempServiceType=Service_type]`，其中：

- `-f` 会强制启动重新创建过程而无需用户确认。如果未使用 `-f` 选项，该命令将提示您确认操作。
- `removeAll` 是可选项，用于删除所有快照以及收件箱内容（上传的文件）和发件箱内容（从环境导出的文件）。默认选项为 `false`，此选项将保留快照以及收件箱和发件箱的内容。
- `TempServiceType`（可选）将一个环境转换为其他服务环境。
  可以在环境中部署的业务流程由您拥有的订阅类型控制。例如，如果您拥有 EPM Standard Cloud Service 订阅，则在将环境从 Account Reconciliation 转换为 Planning 之后不能创建自由形式应用程序。如果您拥有 EPM Enterprise Cloud Service 订阅，则可以在相应地更改服务类型之后在环境中创建任何业务流程。请参阅《管理员入门指南》中的“关于新的云 EPM 服务”

  此参数的行为取决于您的订阅。

  - **除 EPM Standard Cloud Service 和 EPM Enterprise Cloud Service 之外的订阅：**
    可以使用 `TempServiceType` 选项将 Planning、Enterprise Planning、Tax Reporting 或 Financial Consolidation and Close 环境临时转换为 Account Reconciliation、Oracle Enterprise Data Management Cloud 或 Profitability and Cost Management 环境。例如，如果购买了 Planning 环境，则可以通过运行以下命令将其转换为 Account Reconciliation 环境：

    ```
    epmautomate recreate -f removeAll=true TempServiceType=ARCS
    ```

    在将环境转换为 Account Reconciliation 之后，您可以通过使用相应的 `TempServiceType` 值将环境转换为 Oracle Enterprise Data Management Cloud 或

Profitability and Cost Management 环境。例如，要将环境转换为 Profitability and Cost Management，请执行以下命令：

```
epmautomate recreate -f removeAll=true TempServiceType=PCMCS
```

要将环境转换回原始服务类型，请运行以下命令：

```
epmautomate recreate -f
```

Profitability and Cost Management：您可以通过运行以下命令将 Profitability and Cost Management 环境转换为 Planning、Enterprise Planning 或 Enterprise Profitability and Cost Management 环境：

```
epmautomate recreate -f removeAll=true TempServiceType=PBCS
```

要将环境转换回原始 Profitability and Cost Management 环境，请使用以下命令：

```
epmautomate recreate -f TempServiceType=PCMCS
```

> **注：**
>
> Profitability and Cost Management 环境无法转换为 Account Reconciliation、Oracle Enterprise Data Management Cloud 或 Narrative Reporting 环境。

- **EPM Standard Cloud Service 和 EPM Enterprise Cloud Service 订阅：**
  可以使用 `TempServiceType` 选项将 Oracle Fusion Cloud Enterprise Performance Management 环境转换为任何其他受支持的环境。

  EPM Enterprise Cloud Service 订阅使用通用的云 EPM 平台。最初，您可以部署任何受支持的云 EPM 业务流程。

  要从已部署的业务流程切换到其他业务流程，请通过为环境指定新服务类型来重新创建环境。例如，如果已创建 Account Reconciliation 业务流程，但现在希望创建 Oracle Enterprise Data Management Cloud 环境，则必须按如下所示运行重新创建命令。

  ```
  epmautomate recreate -f removeAll=true TempServiceType=EDMCS
  ```

  要将业务流程（例如，Account Reconciliation）转换为 Planning、Tax Reporting 或 Financial Consolidation and Close，则无需指定 `TempServiceType` 值。例如，如果已创建 Account Reconciliation 业务流程，但现在希望创建 Planning 模块环境，则必须按如下所示运行重新创建命令。

  ```
  epmautomate recreate -f removeAll=true
  ```

可接受的 `TempServiceType` 值：

- `ARCS`，用于将环境转换为 Account Reconciliation 环境
- `EDMCS`，用于将环境转换为 Oracle Enterprise Data Management Cloud 环境
- `EPRCS`，用于将环境转换为 Narrative Reporting 环境
- `PCMCS`，用于将环境转换为 Profitability and Cost Management 环境

示例

- 重新创建当前环境，将其还原为原始服务类型（如果之前发出了包含 `TempServiceType` 参数的 recreate），且不删除由用户创建的快照以及收件箱和发件箱内容：

  ```
  epmautomate recreate -f
  ```

- 重新创建当前环境，将其还原为原始服务类型（如果之前发出了包含 `TempServiceType` 参数的 recreate），并删除快照以及收件箱和发件箱内容：

  ```
  epmautomate recreate -f removeAll=true
  ```

- 将当前环境重新创建为 Enterprise Data Management 环境，并删除收件箱和发件箱内容以及现有的快照：

  ```
  epmautomate recreate -f removeAll=true TempServiceType=EDMCS
  ```

- 将当前 EPM Enterprise Cloud Service Account Reconciliation 环境重新创建为 Financial Consolidation and Close 环境，并删除收件箱和发件箱内容以及现有的快照：

  ```
  epmautomate recreate -f removeAll=true
  ```

refreshCube

刷新应用程序多维数据集。通常，在将元数据导入应用程序后刷新多维数据集。

完成多维数据集刷新操作所需的时间取决于您对应用程序结构进行的更改以及该操作对多维数据集的影响。例如，更新稀疏块存储多维数据集成员之后执行刷新操作可能不会花费太多时间，而更新密集块存储多维数据集成员或聚合存储多维数据集成员之后执行多维数据集刷新操作可能会花费大量时间。必须确保在下次维护期间备份应用程序之前已完成多维数据集刷新操作。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate refreshCube [JOB_NAME]`，其中 `JOB_NAME`（可选）是在应用程序中定义的数据库刷新作业的名称。
此操作的状态将回显到运行此命令的控制台。您还可以从应用程序**作业**屏幕的**最近的活动**页面中查看此状态。

示例

```
epmautomate refreshCube DaliyCubeRefresh
```

removeUserFromGroups

从 ANSI 或 UTF-8 编码 CSV 文件中标识的访问控制组中删除用户的成员身份。

文件格式如下：

```
Group Name
Group1
Group2
```

> **注：**
>
> 这些组必须存在于访问控制中。组名的值不区分大小写。

服务管理员使用 uploadFile 命令将文件上传到环境中。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate removeUserFromGroups` *`FILE_NAME User_Login`*，其中：

- `FILE_NAME` 是 CSV 文件的名称，该文件包含要从其中删除用户的成员身份的访问控制组名称
- `User_Login` 是将从访问控制组删除其成员身份的 Oracle Fusion Cloud Enterprise Performance Management 用户的登录 ID。此用户登录 ID 必须存在于为环境提供服务的身份域中，且必须已分配有预定义角色。此值不区分大小写。

示例

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

removeUsers

删除从身份域上传到环境中的 ANSI 或 UTF-8 编码的 CSV 文件中标识的帐户。

文件格式如下：

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

服务管理员使用 uploadFile 命令将文件上传到环境中。用户登录值不区分大小写。例如，`jane.doe@example.com` 被视为与 `Jane.Doe@Example.com` 或其任何大小写变体相同。

> **注：**
>
> - 该 CSV 文件不得包含执行此命令的用户的帐户。
> - 身份域管理员支持的所有服务环境中的用户帐户是公共的，因此删除一个环境中的帐户也将从共享身份域管理员的所有环境中删除该帐户。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

身份域管理员和任何预定义角色

用法

`epmautomate removeUsers FILE_NAME`，其中 *`FILE_NAME`* 是 CSV 文件的名称，该文件包含要从身份域删除的用户的登录 ID。

示例

```
epmautomate removeUsers remove_users.CSV
```

removeUsersFromGroup

从访问控制内保留的组中删除 ANSI 或 UTF-8 编码 CSV 文件中列出的用户。

文件格式如下：

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

用户登录值不区分大小写。例如，`jane.doe@example.com` 被视为与 `Jane.Doe@Example.com` 或其任何大小写变体相同。服务管理员使用 uploadFile 命令将包含用户登录名的文件上传到环境。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate removeUsersFromGroup *FILE_NAME GROUP_NAME*`，其中：

- *FILE_NAME* 是 CSV 文件的名称，该文件包含要从访问控制。内保留的组中删除的用户的登录名
- *GROUP_NAME* 是要从中删除用户的访问控制组的名称。此值不区分大小写。

> **注：**
>
> 仅当下面两个条件都满足时才能从组中删除用户：
>
> - 文件中包括的用户登录名存在于为环境提供服务的身份域中
> - 将为用户分配身份域中的预定义角色

示例

```
epmautomate removeUsersFromGroup user_file.CSV example_group
```

removeUsersFromTeam

从团队中删除 CSV 文件中列出的 Oracle Fusion Cloud Enterprise Performance Management 用户。

如果 CSV 文件中包含的某个用户不是团队成员，此命令将忽略该用户。此文件中的值不区分大小写。CSV 文件格式如下：

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

服务管理员使用 uploadFile 将 .CSV 文件上传到环境中。

适用于

Financial Consolidation and Close、Tax Reporting 和 Account Reconciliation。

所需角色

- 服务管理员
- 任何预定义角色和“团队 - 管理”应用程序角色
- 任何预定义角色和“用户 - 管理”应用程序角色

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate removeUsersFromTeam FILE.CSV TEAM_NAME`，其中：

- `FILE` 确定 UTF-8 格式的 CSV 文件，其中列出要从团队中删除的用户的登录 ID。
- `TEAM_NAME` 确定在访问控制中定义的团队名称。此值不区分大小写。

示例

```
epmautomate removeUsersFromTeam example_users.csv example_team
```

renameSnapshot

重命名在环境中上传或创建的快照。

如果执行此命令以重命名正在生成或存档的快照，您将收到以下错误之一：

- 如果正在生成快照：`File not found`（找不到文件）
- 如果正在对快照进行存档：`Archive process is in progress. Unable to Rename or Delete`（正在存档。无法重命名或删除）

请不要在环境中重命名维护快照。要保留维护快照的备份，应从环境中将 `Artifact Snapshot` 下载到本地计算机，然后根据需要对其进行重命名。请参阅《管理员入门指南》中的“维护快照概览”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate renameSnapshot SNAPSHOT_NAME NEW_SNAPSHOT_NAME`，其中：

- *`SNAPSHOT_NAME`* 是现有快照的名称。此值不应包含特殊字符，例如空格、\（反斜杠）、/（正斜杠）、*（星号）、？（问号）、"（双引号）、<（小于）和 >（大于）。
- *`NEW_SNAPSHOT_NAME`* 是要分配给快照的唯一名称。

示例

```
epmautomate renameSnapshot "Example Snapshot" Example_Snapshot_18_09_25
```

replay

在环境中重放 Oracle Smart View for Office、REST API 或 EPM Automate 负载以在重负载下进行性能测试，从而验证服务在指定负载下时用户体验是否可接受。

必须创建标识了应在服务上执行的活动的重放文件。有关如何创建重放文件的详细信息，请参阅“准备运行 Replay 命令”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

`epmautomate replay` *`REPLAY_FILE_NAME`*`.csv [duration=`*`N`*`] [trace=true] [lagTime=`*`t`*`] [encrypt=true|false]`，其中：

- *`REPLAY_FILE_NAME`* 是存储了要在环境中执行的活动的 CSV 文件。
- `Duration` 为可选参数，指示在环境中执行活动的分钟数。
  如果未设置此值，将对 HAR 文件中的活动运行一次。如果 HAR 文件中的活动运行完成后未到此参数指定的时间，该命令将再次运行 HAR 文件直到活动完成。例如，假设设置 `duration=10` 来重放运行时间为三分钟的 HAR 文件。在此方案中，replay 命令会运行 HAR 文件活动四次（持续 12 分钟），直到第四次迭代完成。
- `trace=true` 是可选设置，指示命令以 XML 格式创建跟踪文件。
  如果指定此可选设置，该命令为重放 CSV 文件中包括的每个 HAR 文件创建一个文件夹并在其中存储所有相关跟踪文件。对于 HAR 文件中的每个活动，此命令生成一个包含 Smart View 响应的跟踪文件。跟踪文件命名为 `trace-`*`N`*`.xml`；例如 `trace-1.xml`，其中 `N` 是从 1 开始的一个计数值。

  将在运行 EPM Automate 的目录中创建存储跟踪文件的文件夹。该命令以 `YYYY_MM_DD_HH_MM_SS_`*`HAR_FILE_NAME`* 格式，组合使用当前环境系统时间与 HAR 文件名来命名文件夹。例如，如果 HAR 文件名是 `forecast1.har`，则文件夹名称可能是 `2016_06_08_10_21_42_forecast1`。
- `[lagTime=`*`t`*`]`（可选）用于指定命令在两次执行重放文件中包含的每个 HAR 文件之间应等待的秒数。默认值为 5 秒。
  如果您指定的值小于 5 秒，该命令将显示错误。负数（例如 -1）和分数（例如 1/2）不能作为此参数的值。支持小数值。

  开始执行第一个 HAR 文件后，命令将等待此参数指定的秒数，再启动对下一个 HAR 文件的处理。由于用户活动通常不是同时启动的，因此设置此参数有助于在环境中创建更真实的负载模拟。

  例如，假设您要模拟高峰期间有 1000 个用户登录到环境执行活动的负载。您可以创建 HAR 文件来模拟这些会话，然后以 6 秒的延时时间运行此命令，以重复环境中存在的负载。
- `encrypt=true|false` 为可选参数，指定是否加密重放文件中包含的所有密码。默认值为 `true`。随机加密密钥用于对密码进行加密。

有关执行此命令涉及的详细步骤，请参阅“重放示例会话”。

示例

```
epmautomate replay forecast1.CSV duration=60 lagTime=5.6
```

resetService

重新启动环境。您可以选择在重新启动环境之前自动优化环境，以确保块存储选项 (BSO) 多维数据集的 Oracle Essbase 索引高速缓存已针对您的应用程序进行了优化。

默认情况下，您的环境在日常维护完成后立即重新启动。自动优化环境非常重要，例如，在将快照导入环境之后。仅当发现严重的性能降级或收到指出环境无法使用的错误消息时，才应使用该命令。重新启动环境不会影响应用程序自定义（例如，区域设置更改、与主题和货币相关的设置等等）。重新启动操作最多需要 15 分钟。

使用此命令前，确保业务规则未在环境中运行。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate resetService "comment" [AutoTune=true|false] [-f]`，其中：

- `Comment` 是针对导致用户重置环境的问题的说明。注释必须用引号括起来。
- `AutoTune`（可选）指示是否自动优化环境以优化应用程序的 Essbase 高速缓存 BSO 多维数据集。默认值为 `false`。

  仅在使用 Essbase BSO 多维数据集的环境中使用此参数：Planning（包括 Planning 模块）、Financial Consolidation and Close 和 Tax Reporting。
- `-f`（可选），指定要在不进行其他用户交互的情况下强制重新启动环境。如果不使用此选项，该命令将提示您确认操作。若要调度使用此命令的脚本，则此选项十分有用。

示例

- `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

复制可用备份快照，以便可以将其导入到环境中。

使用 listBackups 命令确定要还原的备份是否可用。将快照自助还原到环境可以节省您的处理时间。还原快照后，使用 importSnapshot 命令将其导入到环境中。

> **Note:**
>
> 仅限 **Narrative Reporting**：使用此过程导入还原的快照：
>
> 1. 使用 downloadFile 命令将还原的快照下载到本地计算机。
> 2. 将下载的快照重命名为 `EPRCS_Backup`。
> 3. 使用 uploadFile 命令将重命名的快照 (`EPRCS_Backup`) 加载到环境中。确保在命令中指定 `to_be_imported` 上传位置。
> 4. 在 Narrative Reporting 中，打开**日常维护**卡。
>   a. 选择**使用上传的备份快照**。
>   b. 单击**调度还原**以在下次日常维护期间导入快照。
>   c. 单击**是**。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmAutomate restoreBackup` *`SNAPSHOT_NAME`* `[targetName=`*`TARGET_SNAPSHOT_NAME`*`]`，其中：

- *`SNAPSHOT_NAME`* 是 listBackups 命令列出的环境中可用备份快照的名称。
- `targetName`（可选）是目标环境中备份快照的名称，不带扩展名。如果您不指定此值，备份快照将使用 *`SNAPSHOT_NAME`* 还原到目标环境，但使用 _（下划线）替换 /（斜杠）。例如，如果 *`SNAPSHOT_NAME`* 为 `2022-05-14T00:08:56/Artifact_Snapshot.zip`，则 `targetName` 将为 `2022-05-14T00:08:56_Artifact_Snapshot.zip`。

示例

- 除 Narrative Reporting 以外的服务：
  ```
  epmAutomate restoreBackup 2022-05-14T00:08:56/Artifact_Snapshot.zip
  targetName=example_Artifact_Snapshot
  ```
- 仅限 Narrative Reporting：
  ```
  epmAutomate restoreBackup 2022-02-16T21:00:02/EPRCS_Backup
  targetName=Example_EPRCS_Backup
  ```

restructureCube

对块存储多维数据集执行完全重建以消除或减少碎片。重建操作还将删除空块，且不会将应用程序中的任何更改推送到多维数据集。

> **注：**
>
> 运行此命令前，确保无人在使用应用程序。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate restructureCube` *`CUBE_NAME`*，其中 *`CUBE_NAME`* 是多维数据集的名称，与应用程序中的名称完全相同

示例

```
epmautomate restructureCube Plan1
```

roleAssignmentAuditReport

在 OCI（第 2 代）环境中创建审核报表，该报表列出在一段时间（对应为环境指定的审核数据保留期）内对预定义角色和应用程序角色分配所做的更改。默认保留期为 30 天。最多可以将其延长到 90 天，方法是在 Oracle Cloud Identity 控制台中更改**审计保留期（天）**设置。要将审核数据保留 90 天以上，请定期下载并存档此报表和无效登录报表。

角色分配审核报表列出了发生角色更改（在“操作”列中显示）的用户登录名、IDCS 组名或 EPM 组名。它还包括已分配或取消分配的预定义或应用程序角色、执行了角色更改的用户（“执行者”列）以及完成操作时的时间戳（UTC，24 小时格式）。“类型”列指示“名称”列表示什么。它可以有以下三个值之一：“用户”（如果“名称”列标识用户的登录名）、“IDCS 组”（如果“名称”列显示 IDCS 组名）或“EPM 组”（如果“名称”列列出 EPM 组名）。

	A	B	C	D	E	F
1	**Name**	**Type**	**Role**	**Action**	**Performed By**	**Date and Time**
2	IDCS-Group1	IDCS Group	Power User	Assign	serviceadmin@example.com	December 11, 2024 00:47:10 UTC
3	jane.doe@example.com	User	Power User	Assign	serviceadmin@example.com	December 11, 2024 00:47:27 UTC
4	epmIDCSGroup	User	Service Administrator	Assign	serviceadmin@example.com	December 11, 2024 00:48:47 UTC
5	john.doe@example.com	User	Power User	Assign	serviceadmin@example.com	December 11, 2024 00:59:06 UTC
6	john.doe@example.com	User	Ad Hoc - Create	Assign	serviceadmin@example.com	December 11, 2024 01:08:37 UTC
7	john.doe@example.com	User	Access Control - Manage	Unassign	serviceadmin@example.com	December 11, 2024 01:10:30 UTC
8	john.doe@example.com	User	Ad Hoc - Create	Unassign	serviceadmin@example.com	January 07, 2025 03:28:56 UTC

列出以前在环境中分配了预定义角色的已删除用户的信息，并在“名称”列中列出用户的显示名称（名字和姓氏）。在这种情况下，“角色”列指示该用户在其帐户删除之前的预定义角色。此更改不适用于分配给已删除用户的应用程序角色（如果有）；此类分配与用户的用户登录名一起显示。有关示例，请参阅下图中红色框中的信息。

	A	B	C	D	E	F
1	**Name**	**Type**	**Role**	**Action**	**Performed By**	**Date and Time**
2	Jane Doe	User	User	Assign	admin@example.com	January 08, 2025 05:42:47 UTC
3	jane.doe@example.com	User	Run Integrations	Assign	admin@example.com	January 08, 2025 05:44:18 UTC
4	john.smith@example.com	User	Power User	Assign	admin@example.com	January 08, 2025 05:44:18 UTC
5	john.smith@example.com	User	User	Unassign	admin@example.com	January 08, 2025 05:44:18 UTC
6	john.smith@example.com	User	Access Control - View	Assign	admin@example.com	January 08, 2025 05:44:18 UTC
7	epmIDCSGroup	IDCS Group	Viewer	Assign	admin@example.com	January 08, 2025 05:44:22 UTC
8	epmGroup	EPM Group	Ad Hoc - Create	Assign	admin@example.com	January 08, 2025 05:44:22 UTC

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

- 服务管理员
- 任何预定义角色和身份域管理员角色
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmAutomate roleAssignmentAuditReport` *`FROM_DATE TO_DATE FILE_NAME`*`.CSV`，其中：

- *`FROM_DATE`* 指明要生成报表的期间的开始日期（格式为 `YYYY-MM-DD`）。此数据必须位于 Oracle Cloud Identity Console 中指定的审核保留期间内。
- *`TO_DATE`* 指明要生成报表的期间的结束日期（格式为 `YYYY-MM-DD`）。
- *`FILE_NAME`* 是报表的 CSV 文件名。可以使用 downloadFile 命令下载生成的报表。

示例

```
epmAutomate roleAssignmentAuditReport 2024-12-11 2025-01-09 RoleAuditReport.CSV
```

roleAssignmentReport

生成角色分配报表 (.CSV)。

默认情况下，此报表列出预定义角色（例如“服务管理员”）以及分配给用户的应用程序角色（例如 Planning 应用程序角色“审批所有权分配者”、“审批超级用户”、“审批管理员”和“审批流程设计者”）。（可选）还可以生成此报表来列出环境的身份域管理员。此报表与从访问控制生成的角色分配报表的 CSV 版本相匹配。

	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

生成此报表会刷新访问控制中提供的用户和角色信息。

仅适用于 OCI（第 2 代）： Oracle Fusion Cloud Enterprise Performance Management 将停用的用户视为未分配任何预定义角色的用户，即使此类用户在停用后可能具有预定义角色也是如此。此报表中不包括有关停用的用户的信息。

> **注：**
>
> 此命令生成的报表等同于使用 provisionReport 命令生成的报表。

可以使用 downloadFile 命令下载报表。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmautomate roleAssignmentReport` *`REPORT_NAME`*`.CSV [userType=IDAdmins|serviceUsers]`，其中：

- *`REPORT_NAME`* 是报表的名称。
- `userType`（可选）标识要在报表中包含其信息的用户的类型。默认值为 `serviceUsers`。有效值为：
  - `serviceUsers` 将在创建的报表中包含所有功能用户的相关信息（如果没有为身份域管理员分配一个预定义角色以授予应用程序访问权限，则不包含该类用户的相关信息）。
  - `IDAdmins` 将在创建的报表中仅列出分配了身份域管理员角色的用户。

示例

- 生成仅列出功能用户的报表：
  - `epmautomate roleAssignmentReport myReport.CSV`

  - `epmautomate roleAssignmentReport myReport.CSV userType=serviceUsers`
- 生成仅列出身份域管理员的报表：
  `epmautomate roleAssignmentReport myReport.CSV userType=IDAdmins`

runAutomatch

使用服务管理员定义的规则运行自动匹配流程来匹配事务。

> **注：**
>
> 在使用 importTmPremappedTransactions 或 runDataRule 命令将事务数据导入事务匹配后再运行此命令。

您可以在 Account Reconciliation 中的**作业历史记录**选项卡上监视自动匹配流程的状态。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate runAutomatch RECONCILIATION_TYPE`，其中 `RECONCILIATION_TYPE` 是在 Account Reconciliation 中定义的调节类型。

示例

```
epmautomate runAutomatch INTERCOMPANY
```

runBatch

执行数据管理批处理。

如果数据管理中的批处理执行模式设置为串行，则会在完成批处理中的所有作业时返回控制；如果模式设置为并行，则会在提交批处理中的所有作业准备执行时返回控制。

> **注：**
>
> 此命令无法用于执行从数据源到 Oracle Fusion Cloud Enterprise Performance Management 的直接数据加载集成。使用 EPM 集成代理来集成直接数据加载。有关详细信息，请参阅《管理数据集成》中的“使用 EPM 集成代理执行直接数据加载”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate runBatch BATCH_NAME`，其中 *BATCH_NAME* 是在数据管理中定义的批处理的名称。

示例

```
epmautomate runBatch Accounting_batch
```

runBusinessRule

启动业务规则。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户（如果授予对规则的启动访问权限）

用法

`epmautomate runBusinessRule RULE_NAME [PARAMETER=VALUE]`，其中：

- *RULE_NAME* 是业务规则的名称，与在环境中定义的完全一样。
- *PARAMETER=VALUE* 表示可选运行时参数以及执行业务规则所需的参数值。

> **注：**
>
> – 此命令只能执行单个业务规则。要执行规则集，请使用 runRuleSet 命令。
>
> – 将根据规则部署到的规划类型执行规则。
>
> – 如果您没有为运行时参数提供值，则使用默认值。该命令忽略与规则中定义的提示不完全匹配的运行时提示。
>
> – 使用 *PARAMETER=VALUE* 对可以指定业务规则所需的任意数量的运行时提示。以下示例使用两个运行时提示（`Period` 和 `Entity`）及其值（`Q1` 和 `USA`）。有关为一个参数输入多个值的信息，请参阅“为一个参数指定多个值”。

示例

```
epmautomate runBusinessRule RollupUSSales Period=Q1 Entity=USA
```

runCalc

在应用程序中运行计算。

使用此命令，可以使用模型 POV 中的规则对不同数据 POV 中的数据运行计算，而无需跨 POV 复制规则。

适用于

Profitability and Cost Management

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate runCalc` *`APPLICATION_NAME POV_NAME [DATA_POV_NAME]`* `PARAMETER=`*`VALUE`* `[comment="comment"] stringDelimiter="`*`DELIMITER`*`"`，其中：

- *`APPLICATION_NAME`* 是包含要计算 POV 的 Profitability and Cost Management 应用程序的名称。
- *`POV_NAME`* 是要计算的模型 POV 的名称。
- *`DATA_POV_NAME`* 为可选，是要使用模型 POV 的规则计算的数据 POV 的名称。

  如果未指定 *`DATA_POV_NAME`*，默认情况下，将使用 *`POV_NAME`*。

  如果指定了 *`DATA_POV_NAME`*，则只能使用 `exeType=ALL_RULES`。
- `PARAMETER=`*`VALUE`* 指明用于运行计算的运行时参数及其值。指定进程所需数量的参数和值对。有效参数及其值：
  - `exeType=ALL_RULES|RULESET_SUBSET|SINGLE_RULE` 标识规则执行类型。这是一个必需的参数。
    根据为 `exeType` 设置的值，可以指定以下参数：
    * 如果指定 `exeType=ALL_RULES`，请勿包含规则子集或单个规则相关的参数，例如 `subsetStart`、`subsetEnd`、`ruleSetName` 和 `ruleName`。如果设置 *`DATA_POV_NAME`* 参数，必须使用此 `exeType`。
    * 如果指定 `exeType=SINGLE_RULE`，请仅为 `ruleSetName` 和 `ruleName` 指定值。
    * 如果指定 `exeType=RULESET_SUBSET`，请为 `subsetStart` 和 `subsetEnd` 指定值。
  - `subsetStart` 指定要运行的规则集中第一个规则的序号
  - `subsetEnd` 指定要运行的规则集中最后一个规则的序号
  - `ruleSetName` 标识要运行的计算所在的规则集
  - `ruleName` 要运行的规则的名称（运行单个规则）
  - `isClearCalculated=true|false` 指定是否清除现有计算
  - `isExecuteCalculations=true|false` 指定是否运行计算

  - `isRunNow=true|false` 将此值设置为 `true` 时，将立即运行进程
  - `optimizeReporting=true|false` 将此可选值设置为 `false`（如果要运行计算但不针对报表进行优化）。默认值为 `true`
    最佳做法：
    * 仅在必要时设置 `optimizeReporting=false` 以节省处理时间；例如，在运行单个规则或连续系列的多个 POV 时
    * 运行多个并发的计算作业时，为所有作业设置 `optimizeReporting=true`；只有最后一个要完成的作业将执行聚合，这样可以避免冗余处理并防止作业的运行速度下降。

> **注：**
>
> 参数值（`true` 或 `false`）必须全部小写。

- `comment` 是用双引号括起来的可选注释
- `stringDelimter` 是 POV 值中使用的分隔符。分隔符必须用双引号括起来。

示例

```
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

生成在“调节合规性”中定义的报表。

请参阅《管理 *Account Reconciliation*》中的以下信息源：

- 使用报表，以获得定义报表的说明。
- 在调节合规性中生成预定义报表，以获得预定义“调节合规性”报表的列表及用于生成这些报表的参数。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate runComplianceReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]`，其中：

- FILE_NAME 是要生成的报表的唯一文件名。如果服务器上已存在具有此名称的报表，它将被覆盖。使用 downloadFile 命令将此报表下载到本地计算机。
- GROUP_NAME 是报表所关联的组的名称。
- REPORT_NAME 是要生成的报表的唯一名称。
- REPORT_FORMAT 是以下报表格式之一：
  - PDF
  - HTML（不支持图形和图表）
  - XLSX（不支持图形）
  - CSV
  - CSV2

> **注：**
>
> *REPORT_FORMAT* CSV 不允许根据模板格式化数据，而 CSV2 则允许。与 CSV 输出相比，生成 CSV2 格式报表所需的时间更长。

- Param=value（可选），标识生成报表所需的参数。例如，按帐户类型的余额报表采用以下两个参数值：值为 July 2017 的 Period 以及值为 Entered 的 Currency Bucket。您应将这些参数指定为 "Period=July 2017" "Currency Bucket=Entered"。

示例

```
epmautomate runComplianceReport "Example_File Name""Reconciliation Manager"
"Balance By Account Type" PDF "Period=July 2017" "Currency Bucket=Entered"
```

runDailyMaintenance

立即启动日常服务维护流程，而不是等待调度的日常维护期间。

使用此命令可强制创建备份快照以及更新环境。运行此命令前，确保无人在使用环境。日常维护调度不受此命令的影响。如果不想等到下次维护期间而想让环境更改立即生效（例如，应用一次性修补程序后），则使用此命令。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员、用户（仅 Oracle Enterprise Data Management Cloud）

用法

epmautomate runDailyMaintenance [skipNext=true|false] [-f]，其中：

- skipNext 为可选，用于指示是否跳过下一次的日常维护流程。默认值为 false。
- -f 为可选，用于指示是否强制启动维护流程而无需用户确认。如果未使用 -f 选项，该命令将提示您确认操作。

示例

- 要强制启动非周期日常维护而不跳过下一个调度的维护：`epmautomate runDailyMaintenance -f`
- 要强制启动非周期日常维护并跳过下一个调度的维护：`epmautomate runDailyMaintenance skipNext=true -f`
- 要启动非周期日常维护并跳过下一个调度的维护：`epmautomate runDailyMaintenance skipNext=true`

runDataRule

基于指定的起始期间和结束期间以及导入或导出选项执行数据管理数据加载规则。

> **注：**
>
> 此命令无法用于执行从数据源到 Oracle Fusion Cloud Enterprise Performance Management 的直接数据加载集成。使用 EPM 集成代理来集成直接数据加载。有关详细信息，请参阅《管理数据集成》中的“使用 EPM 集成代理执行直接数据加载”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate runDataRule` *`RULE_NAME START_PERIOD END_PERIOD IMPORT_MODE EXPORT_MODE [FILE_NAME]`*，其中：

- *`RULE_NAME`* 是数据管理中定义的数据加载规则的名称。如果规则名称中包含空格，则应该使用引号将其括起来。
- *`START_PERIOD`* 是要加载数据的第一个期间。此期间名称必须在数据管理期间映射中进行定义。
- 对于多期间数据加载，*`END_PERIOD`* 是要加载数据的最后一个期间。对于单期间加载，请输入与起始期间相同的期间。此期间名称必须在数据管理期间映射中进行定义。
- *`IMPORT_MODE`* 用于确定如何将数据导入到数据管理。

  导入模式设置区分大小写。可接受的值包括：

  - `APPEND`，用于添加到数据管理中的现有 POV 数据
  - `REPLACE`，用于删除 POV 数据并将其替换为文件中的数据
  - `RECALCULATE`，用于重新计算数据
  - `NONE`，用于跳过将数据导入到数据管理临时表的操作
- *`EXPORT_MODE`* 用于确定如何将数据导出到应用程序。

导出模式设置区分大小写。可接受的值包括：

- STORE_DATA，用于合并数据管理临时表中的数据和现有数据。始终在用于加载元数据的数据管理作业中使用此导出选项。
- ADD_DATA，用于将数据管理临时表中的数据添加到应用程序。
- SUBTRACT_DATA，用于从现有数据中减去数据管理临时表中的数据。
- REPLACE_DATA，用于清除 POV 数据并将其替换为数据管理临时表中的数据。“方案”、“版本”、“年”、“期间”和“实体”中的数据将被清除。
- NONE，用于跳过从数据管理到应用程序的数据导出。

> **注：**
>
> 对于 Financial Consolidation and Close，仅支持以下导出模式：
>
> - MERGE，用于合并数据管理临时表中的数据和现有数据
> - REPLACE，用于删除 DM 临时表中的条目并将其替换为数据加载中的条目
> - NONE，用于跳过从数据管理到应用程序的数据导出
>
> 如果 Oracle Fusion Cloud 作为目标，仅支持以下导出模式：
>
> - MERGE，用于合并数据管理临时表中的数据和现有数据
> - NONE，用于跳过从数据管理到应用程序的数据导出

- *FILE_NAME* 是可选的文件名。如果未指定文件名，则该命令会导入加载数据规则指定的文件名中包含的数据。此文件必须位于收件箱文件夹中或该文件夹内的某个文件夹中。

  为 Account Reconciliation 加载银行管理协会 (Bank Administration Institute, BAI) 格式的文件时，不要指定此参数的值。必须始终在数据规则定义中指定用于加载 BAI 文件的文件名。

> **注：**
>
> 如果在数据规则中指定了路径，则不要在命令中指定文件路径；请仅指定文件名。如果未在路径规则中指定路径，请指定数据文件的完整路径。

示例

- 多个期间导入：

```
epmautomate runDataRule VisionActual Mar-15 Jun-15 REPLACE STORE_DATA inbox/
Vision/GLActual.dat
```

- 单个期间导入：

```
epmautomate runDataRule "Vision Actual" Mar-15 Mar-15 REPLACE STORE_DATA
inbox/Vision/GLActual.dat
```

runDMReport

创建数据管理报表并将其存储在 outbox/reports 文件夹中。

生成的报表基于生成报表的数据管理作业的 ID 和报表格式进行命名。例如，如果报表作业 ID 为 2112 并且您指定的报表输出格式为 PDF，则报表名称为 2112.pdf。生成报表后，报表名称

将显示在控制台中。您还可以从数据管理中的“进程详细信息”选项卡或通过使用 listFiles 命令来识别报表名称。

使用 downloadFile 命令将该报表下载到本地计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

`epmautomate runDMReport REPORT_NAME PARAMETER=Value "Report Output Format=[PDF|HTML|XLS|XLSX]"`，其中：

- *REPORT_NAME* 是用于生成报表的数据管理报表模板的名称。
- *PARAMETER=Value* 指示报表参数及其值。以 PARAMETER=Value 格式指定所需数量的参数。所需参数的列表取决于要生成的报表。

> **注：**
>
> 设计报表时定义报表运行时参数。要运行此命令，您必须从“工作流”选项卡生成并使用这些参数和值。要生成报表的运行时参数，请在数据管理的“工作流”选项卡中单击**报表执行**，然后从**报表组**中选择组。选择要为其生成参数的报表，然后单击**创建报表脚本**。（可选）指定报表参数值，然后选择输出格式，再单击**确定**。使用**生成报表脚本**中显示的参数指定运行时参数和值来生成报表

- Report Output Format 指示报表输出格式。有效选项为 PDF、HTML、XLS 和 XLSX。默认报表格式为 PDF。

示例

```
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

执行数据集成作业以将数据导入到 Oracle Fusion Cloud Enterprise Performance Management 业务流程，或将数据从业务流程导出到外部系统。

使用此命令将弃用 runDataRule 命令。Oracle 建议您开始使用此命令而非 runDataRule 命令。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户

用法

```
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]
```

- 对于标准模式集成，您必须指定 `importMode`、`exportMode` 和 `periodName` 的值
- 对于快速模式集成，您必须指定 `exportMode` 的值
- 参数名称及其值区分大小写

在此命令中：

- *JOB_NAME* 是在数据集成中定义的集成作业的名称。
- `importMode` 确定如何将数据导入到数据集成。可接受的导入模式包括：
  - `Append`，用于添加到数据集成中的现有 POV 数据。
  - `Replace`，用于删除 POV 数据并将其替换为文件中的数据。
  - `Map and Validate`，用于跳过数据导入，但使用更新的映射和逻辑帐户重新处理数据。
  - `No Import`，用于跳过将数据导入到数据集成临时表的操作。
- `exportMode` 确定如何将数据加载到目标应用程序。对于快速模式集成，不能使用 `Check` 和 `No Export` 作为 `exportMode` 参数的值。可接受的导出模式值包括：
  - `Merge`，用于更新现有数据并添加新数据。
  - `Replace`，用于清除 POV 中的现有数据并为其加载新数据。对于标准模式，“方案”、“版本”、“年”、“期间”和“实体”维中的数据将被清除。对于快速模式，“年”、“期间”和“实体”维中的数据将被清除。您可以为这两种模式定义自定义清除区域。
  - `Accumulate`，用于将数据添加到现有数据。适用于 Planning、Planning 模块、Financial Consolidation and Close、Tax Reporting、Profitability and Cost Management 和 Enterprise Profitability and Cost Management。
  - `Subtract`，用于从现有余额中减去数据。适用于 Profitability and Cost Management 和 Enterprise Profitability and Cost Management。
    对于快速模式集成：
    * 不能使用 `Check` 和 `No Export` 作为此参数的值。
    * 对于 Planning、Planning 模块和 Financial Consolidation and Close，只有以下值有效：`Replace`、`Merge` 和 `Accumulate`。
  - `No Export`，用于跳过数据导出。使用此模式，可在数据加载到目标应用程序之前，先将数据加载到临时表中以供查看。
  - `Check`，用于仅执行数据验证检查。

    如果 Oracle Fusion Cloud 作为目标，仅支持以下导出模式：
    * `MERGE`，用于合并数据集成临时表中的数据和现有数据
    * `NONE`，用于跳过从数据集成到应用程序的数据导出

- periodName 是一个或多个期间或期间范围的名称，每个期间或期间范围都用大括号括起，用于导入或导出数据。可接受的期间命名惯例如下：
  - 对于单期间加载，指定用大括号括起的期间名称，例如 {Jan-21}
  - 对于多个期间加载，将开始和结束期间名称括在大括号中，例如 {Jan-21}{Mar-21}（用于加载从 1 月 21 日开始到 3 月 21 日结束的所有期间的数据）
  - **对于 Planning、Planning 模块、Financial Consolidation and Close、自由形式和 Tax Reporting：**可以指定业务流程期间名称和格式为 {Jan#FY21}{Mar#FY21} 的年份，以加载从 1 月 21 日开始到 3 月 21 日结束的所有期间的数据。
    期间名称必须用大括号括起。
    * 单个期间 - 是指期间映射中定义的单个期间的数据管理期间名称。
    * 多个期间 - 是指多个期间加载。该参数以 {Month-Year}{Month-Year} 格式指定。例如，{Jan-20}{Mar-20} 表示从 1 月 20 日到 3 月 20 日的多个期间加载。
    * Planning 期间名称 - 是指 {Month#Year} 格式的 Planning 期间名称，例如 {Jan#FY20}{Mar#FY20}。使用此惯例，您无需指定数据集成期间名称，而是指定“年”和“方案”维的成员名称。
      Planning、Tax Reporting 和 Financial Consolidation and Close 业务流程支持此参数。它适用于您的服务应用程序和从内部部署数据源派生的云部署。

      如果通过捕获“年”和“期间”成员名称从云 EPM Groovy 脚本触发，则使用此惯例非常有用。应用程序期间映射或全局期间映射必须存在期间映射的目标值中的“年”和“月”。
    * 替代变量 - 这是上述 Planning 期间名称格式的扩展，其中可以按 {Month#&CurYr}{&FcstMonth#&CurYr} 格式指定替代变量，而不是实际的“年”和“月”成员名称，例如 {Jan#&CurYr}{&FcstMonth#&CurYr}。
      支持实际成员名称和替代变量的组合。

      Planning、Tax Reporting 和 Financial Consolidation and Close 业务流程支持此格式。

      应用程序期间映射或全局期间映射必须存在于从中运行命令的环境的数据集成中，并且期间映射的目标值中存在“年”和“月”值在这种情况下，“年”和“月”是指执行期间替代变量的当前值。
    * GLOBAL POV - 执行全局 POV 期间的数据加载。使用格式 {GLOBAL_POV}。

    > **Note:**
    > 如果您使用除本次讨论中所述参数以外的任何期间命名参数，您将收到 Invalid Input – HTTP 400 错误消息。
  - {GLOBAL_POV}，用于在系统的全局 POV 中或在数据集成的应用程序设置中定义的期间内执行数据加载。

> **Note:**
>
> Planning、Planning 模块、Financial Consolidation and Close 和 Tax Reporting 支持 {Month#Year} 期间命名惯例格式。根据此惯例，您可以为“年”和“方案”维指定成员名称，而非数据集成期间名称。如果命令是通过捕获“年”和“期间”成员名称从 Groovy 脚本触发的，则此方法很有用。`{Jan#&CurYr}{&FcstMonth#&CurYr}` 替代变量命名惯例是前一期间命名惯例的扩展。如果对 Planning、Planning 模块、Financial Consolidation and Close 和 Tax Reporting 运行此命令，则可以指定替代变量而不是“年”和“月”成员名称。同样支持成员名称和替代变量的组合。
>
> 仅当数据集成中已经存在目标值包含“年”和“月”的应用程序期间映射或全局期间映射时，前一期间命名惯例和替代变量命名惯例才有效。

- `inputFileName`，（对于基于文件的数据加载）用于指定要从中导入数据的收件箱中可用的文件名称。如果您在集成定义中指定目录名称，则仅传递文件名。如果不在集成定义中包含目录名称，请使用 `inbox/DIR_NAME/FILE_NAME` 格式，例如 `inbox/GLBALANCES.txt` 或 `inbox/EBSGL/GLBALANCES.txt`。如果文件已上传到环境中的默认位置，请使用 `#epminbox/FILE_NAME` 惯例（例如 `#epminbox/GLBALANCES.txt`）标识输入数据文件。
  此参数仅适用于基于本机文件的数据加载。如果不为基于文件的数据加载指定此参数值，则此命令会从集成定义中指定的文件导入数据。如果您为不基于文件的数据加载指定此参数值，该命令会将其忽略。
- *PARAMETERS*（可选）标识 *PARAMETER_NAME="PARAMETER"* 格式的运行时参数。参数包括源筛选器和目标选项。

> **Note:**
>
> 此时您可以对目标应用程序的维（元数据）类型使用的唯一参数是 `"Refresh Database"=Yes|No`。

示例

- 单个期间导入：
  ```
  epmAutomate runIntegration VisionDataLoad importMode=Replace exportMode=Merge
  periodName="{Mar-15}"
  ```
- 多个期间导入：
  ```
  epmAutomate runIntegration VisionDataLoad importMode=Replace exportMode=Merge
  periodName="{Mar-15}{Jun-15}"
  ```
- 基于增量文件的数据集成：
  ```
  epmAutomate runIntegration IncrementalFileLoad importMode=Replace
  exportMode=Merge periodName="{Jan-20}{Mar-20}" inputFileName=File1.txt
  ```

runIntercompanyMatchingReport

生成公司内匹配报表，帮助您在合并过程中以及出于分析和审核目的正确匹配相关实体和合作伙伴之间的事务。

适用于

Financial Consolidation and Close

所需角色

- 服务管理员
- 超级用户（如果通过 ACL 授予额外安全性）
- 用户（如果通过 ACL 授予额外安全性）

用法

`epmautomate runIntercompanyMatchingReport JOB_NAME FILE_NAME [scenario=scenario] [years=years] [period=period_name] [ReportFormat=HTML]`，其中：

- `JOB_NAME` 是应用程序中存在的公司内报表作业定义的名称。有关详细信息，请参阅《使用 *Financial Consolidation and Close*》中的“管理公司内匹配报表”。此命令使用作业中的可用设置生成公司内匹配报表。确保选择发件箱作为生成的报表的位置。

  > **Note:**
  >
  > 您可以通过在运行此命令时输入方案、年份、期间和报表格式设置来覆盖作业中指定的这些可选值。

- `FILE_NAME` 是报表文件的名称。将在收件箱中生成此报表。使用 downloadFile 命令将其下载到本地计算机，或者使用 sendMail 命令将其通过电子邮件发送。
- `SCENARIO`（可选）是应用程序中定义的方案名称，将为其生成报表。
- `YEARS`（可选）是将为其生成报表的年份。
- `PERIOD`（可选）是将为其生成报表的期间。
- `ReportFormat`（可选）是报表的格式。可接受的值为 HTML、PDF 和 XLSX。

> **Note:**
>
> - 您可以为一个方案、年份和期间的组合生成此报表。不要指定多个方案、年份或期间。
> - 如果您没有指定方案、年份、期间和报表格式的值，则使用报表作业定义（由 `JOB_NAME` 标识）中指定的相应值。如果指定，这些可选值将覆盖报表作业定义中设置的值。

示例

- 通过覆盖作业定义中设置的值来创建报表：
  ```
  epmautomate runIntercompanyMatchingReport IC_Job_01 SampleICReport.html
  scenario=Actual years=FY22 period=Jan reportFormat=HTML
  ```
- 使用作业定义中设置的值创建报表：
  ```
  epmautomate runIntercompanyMatchingReport IC_Job_01 SampleICReport
  ```

runMatchingReport

生成在“事务匹配”中定义的报表。

有关预定义“事务匹配”报表的列表及用于生成这些报表的参数，请参阅《管理 *Account Reconciliation*》中的“在事务匹配中生成预定义报表”。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate runMatchingReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]`，其中：

- `FILE_NAME` 是要生成的报表的唯一文件名。如果服务器上已存在具有此名称的报表，它将被覆盖。使用 downloadFile 命令将此报表下载到本地计算机。
- `GROUP_NAME` 是报表所关联的组的名称。
- `REPORT_NAME` 是要生成的报表的唯一名称。
- `REPORT_FORMAT` 是以下报表格式之一：
  - PDF
  - HTML（不支持图形和图表）
  - XLSX（不支持图形）
  - CSV
  - CSV2

> **注：**
>
> *`REPORT_FORMAT`* `CSV` 不允许根据模板格式化数据，而 `CSV2` 则允许。与 `CSV` 输出相比，生成 `CSV2` 格式报表所需的时间更长。

- `Param=Value`（可选），标识生成报表所需的参数。例如，对于采用值为 `approved` 的参数 `status` 的匹配类型配置报表，应将参数和值指定为 `status=Approved`。

示例

```
epmautomate runMatchingReport Example_FileName "Transaction Matching" "Match Type
Configuration" HTML "status=Approved"
```

runPipeline

根据为管道定义的变量运行数据集成管道（将一系列作业编排为单个流程）。有关管道的详细信息，请参阅《在 *Oracle Enterprise Performance Management Cloud* 中管理数据集成》中的“使用管道”。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Enterprise Profitability and Cost Management、Tax Reporting、销售规划和战略性人员规划

所需角色

服务管理员或任何设置为代理用户以运行管道定义（通过使用位置安全性或将系统设置的**为非管理用户启用管道执行**设置指定为是）的用户。

用法

`epmautomate runPipeline` *`PIPELINE_CODE`* `[PARAMETER=VALUE]`，其中：

- *`PIPELINE_CODE`* 是在数据集成中创建管道时定义的代码。有关详细信息，请参阅《管理数据集成》中的“管道进程说明”。
- `PARAMETER=`*`VALUE`*（可选）标识运行管道的参数及其值。指定进程所需数量的参数和值对。参数列表取决于数据集成的“管道变量”屏幕中定义的变量数。默认管道参数和可接受的值：
  - `STARTPERIOD`：要加载数据的第一个期间。此期间名称必须在数据集成期间映射中进行定义。
  - `ENDPERIOD`：要加载数据的最后一个期间。此期间名称必须在数据集成期间映射中进行定义。
  - `IMPORTMODE` 确定如何将数据导入到数据集成。可接受的值包括：
    - `Append` 用于添加到数据集成中的现有 POV 数据。
    - `Replace` 删除 POV 数据并将其替换为管道定义中指定或作为参数的文件中的数据。
    - `Map and Validate` 使用更新的映射和逻辑帐户重新处理数据，而无需从管道定义中指定或作为参数的文件中导入数据。
    - `No Import` 跳过将数据导入到数据集成临时表的操作。
  - `EXPORTMODE` 确定如何将数据导出到数据集成。可接受的导出模式包括
    - `Merge` 合并数据集成临时表中的数据与业务流程中的现有数据。
    - `Accumulate` 将数据集成临时表中的数据添加到业务流程。
      Planning、Planning 模块、自由形式、销售规划和战略性人员规划。
    - `Replace` 删除业务流程中的 POV 数据并将其替换为数据集成临时表中的数据。
      Scenario、Version、Year、Period 和 Entity 维中的数据将被清除。
    - `No Export` 跳过将数据从数据集成导出到业务流程中的操作。
  - `ATTACH_LOGS` 指定是否将日志文件压缩并附加到与管道执行相关的通知中。可接受的值为 `Y`（表示“是”）和 `N`（表示“否”）。
  - `SEND_MAIL` 指定是否在管道运行完成时发送电子邮件。可接受的值为：`Always`、`No`（默认值）、`On Failure` 和 `On Success`。
  - `SEND_TO` 是要将电子邮件发送到的电子邮件 ID 的逗号分隔列表。

示例

```
epmautomate runPipeline DAILYLOAD "STARTPERIOD=Jan-24" "ENDPERIOD=Jan-24"
"IMPORTMODE=Replace" "EXPORTMODE=Merge" "SEND_MAIL=Always"
"SEND_TO=John.Doe@example.com, Jane.Doe@example.com" "ATTACH_LOGS=Y"
```

runPlanTypeMap

根据在 `plan type map` 类型的作业中指定的设置，将数据从一个块存储数据库复制到一个聚合存储数据库或者从一个块存储复制到另一个块存储。

适用于

Planning,、Planning 模块、自由形式、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate runPlanTypeMap` *`JOB_NAME`* `[clearData=true|false]`，其中：

- `JOB_NAME` 是在应用程序中定义的 `plan type map` 类型的作业的名称。
- `clearData` 是一项可选设置，指明是否应该在复制数据之前删除目标数据库中的数据。如果未设置该参数值，将使用默认值 `true`。

  参数值（`true` 或 `false`）必须全部小写。

示例

```
epmautomate runPlanTypeMap CampaignToReporting clearData=false
```

runRuleSet

启动业务规则集。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员、超级用户（如果授予规则启动访问权限）

用法

`epmautomate runRuleSet` *`RULESET_NAME`* `[`*`PARAMETER=VALUE`*`]`，其中：

- *`RULESET_NAME`* 是业务规则集的名称，与在环境中定义的完全一样。
- *`PARAMETER=VALUE`* 表示执行规则集所需的可选运行时参数及其值。

> **注：**
>
> 将根据规则集部署到的规划类型执行规则集。
>
> 使用 *PARAMETER=VALUE* 对可以指定规则集所需的任意数量的运行时提示。以下示例使用两个运行时提示（`Period` 和 `Entity`）及其值（`Q1` 和 `USA`）。
>
> 如果您没有为运行时参数提供值，则使用默认值。该命令忽略与为规则集定义的提示不完全匹配的运行时提示。
>
> 有关为一个参数输入多个值的信息，请参阅“为一个参数指定多个值”。

示例

```
epmautomate runRuleSet RollupUSSales Period=Q1 Entity=USA
```

runSupplementalDataReport

生成显示 Supplemental Data Manager 中数据的关系报表。

补充数据报表在 Financial Consolidation and Close 和 Tax Reporting 中归组为非合并报表。有关可生成的报表的列表以及用于生成这些报表的参数，请参阅 《*REST APIs*》指南中的 "Generate Report for Financial Consolidation and Close and Tax Reporting" 的 "List of Predefined Reports and Parameters" 部分。

适用于

Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

`epmautomate runSupplementalDataReport` *`FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT`* `[Param=value]`，其中：

- `FILE_NAME` 是报表的唯一文件名。
- `GROUP_NAME` 是报表所关联的组的名称。
- `REPORT_NAME` 是要生成的报表的唯一名称。
- `REPORT_FORMAT` 是以下报表格式之一：
  - PDF
  - HTML（不支持图形和图表）
  - XLSX（不支持图形）
  - CSV
  - CSV2

REPORT_FORMAT CSV 不允许根据模板格式化数据，而 CSV2 则允许。与 CSV 输出相比，生成 CSV2 格式报表所需的时间更长。

- Param=value（可选），标识生成报表所需的参数。例如，要生成 schedule name 值为 monthly、period 值为 Jan 的 At Risk Tasks 报表，可指定 "schedule name"=monthly period=Jan。

示例

```
epmautomate runSupplementalDataReport Example_File_name Group1 "At Risk Tasks"
html "schedule name"=monthly period=Jan
```

runTaskManagerReport

生成显示任务管理器中数据的关系报表。

适用于

Enterprise Profitability and Cost Management、Financial Consolidation and Close 和 Tax Reporting。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

epmautomate runTaskManagerReport *FILE_NAME GROUP_NAME REPORT_NAME REPORT_FORMAT* [Param=value]，其中：

- FILE_NAME 是报表的唯一文件名。
- GROUP_NAME 是报表所关联的组的名称。
- REPORT_NAME 是要生成的报表的唯一名称。
- REPORT_FORMAT 是以下报表格式之一：
  - PDF
  - HTML（不支持图形和图表）
  - XLSX（不支持图形）
  - CSV
  - CSV2

> **注：**
>
> *REPORT_FORMAT* CSV 不允许根据模板格式化数据，而 CSV2 则允许。与 CSV 输出相比，生成 CSV2 格式报表所需的时间更长。

- `Param=value`（可选），标识生成报表所需的参数。例如，要生成 `schedule name` 值为 `monthly`、`period` 值为 `Jan` 的 `Early Tasks` 报表，可指定 `"schedule name"=monthly period=Jan`。

示例

```
epmautomate runTaskManagerReport Example_File_name Group1 "Early Tasks" PDF
"schedule name"=monthly period=Jan
```

sendMail

使用可附加 Oracle Fusion Cloud Enterprise Performance Management 文件的选项发送电子邮件。

可以将此命令加入到脚本中，以向用户告知各种情况或者发送报表。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Narrative Reporting、Oracle Fusion Cloud Enterprise Data Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate sendMail ToAddress Subject [Body="MessageBody"] [Attachments=FILE1,FILE2]`，其中：

- `ToAddress` 标识用分号分隔的收件人电子邮件地址，并用双引号将整个字符串括起来。示例：`"jdoe@example.com;jane.doe@example.com"`。
- `Subject` 标识电子邮件主题。
- `Body="MessageBody"`（可选）是电子邮件内容。如果未指定，则电子邮件没有正文。

> **Note:**
>
> 使用有效 HTML 标记设置消息正文的格式，以创建所需的电子邮件格式。整个消息正文（包括所有 HTML 标记）必须在一行中指定，并且不得包含换行符。请参阅“示例”。

- `Attachments`（可选）标识云 EPM 中要附加到电子邮件的可用文件的逗号分隔列表。例如，`outbox/errorFile.txt,inbox/users.csv`。

> **Note:**
>
> – 使用 *（星号）作为文件名中单个字符的通配符。例如，指定 `outbox/user*.csv` 以附加发件箱中文件名（含五个字母）符合此模式的所有文件，
>
> – 可以将 listFiles 命令列出的任何文件（快照除外）附加为电子邮件附件。附件大小不得超过 10 MB。

示例

- **未设格式的电子邮件：** `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`
- **设置格式的电子邮件：** `epmautomate sendMail jdoe@example.com "Send Formatted Email" "Body=<!DOCTYPE html> <html> <head> <title>EpmAutomate Email Formatting</title> </head> <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

将应用程序置于管理模式，以仅限服务管理员可以访问该应用程序。

要在服务管理员正在执行管理操作时防止其他用户使用该应用程序，该命令十分有用。应用程序将一直处于管理模式，直至您重新做出更改后才可供所有用户访问。

> **Note:**
>
> 此命令替代已废弃的 applicationAdminMode 命令，但后者尚未从 EPM Automate 中删除。

使用 getApplicationAdminMode 命令检查环境的当前状态。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、Account Reconciliation、战略性人员规划和销售规划。

所需角色

服务管理员

用法

```
epmautomate setApplicationAdminMode true|false
```

在此命令中，指定 `true` 将应用程序置于管理模式，指定 `false` 将应用程序恢复为正常模式，以供所有用户访问

示例

- 将应用程序置于管理模式：
  ```
  epmautomate setApplicationAdminMode true
  ```
- 将应用程序恢复为正常运行：
  ```
  epmautomate setApplicationAdminMode false
  ```

setDailyMaintenanceStartTime

设置环境日常维护开始时间（UTC 或其他时区）。环境维护不必在整点开始；您可以设置应开始维护的小时和分钟。

为确保使用此命令不影响 Oracle 对备份创建的要求，如果日常维护流程在过去 36 个小时内未运行，此命令将不更改维护开始时间。

> **注：**
>
> 当前使用浏览器登录到环境的服务管理员只有在先注销再登录后，才能看到新的日常维护开始时间。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员、用户（仅 Oracle Enterprise Data Management Cloud）、任何预定义角色和“迁移 - 管理”应用程序角色

用法

`epmautomate setDailyMaintenanceStartTime` *`StartTime`*，其中 *`StartTime`* 是应当启动维护流程的时间（24 小时制，格式为 HH:MM）和可选的时区。可接受的开始时间值范围为 00:00 - 23:59。如果未以 UTC 格式设置开始时间，请指定有效的标准时区；例如，`"14:35 America/Los_Angeles"` 表示太平洋标准时间下午 2:35。

示例

- 将日常维护开始时间设置为 2:20 PM UTC：
  ```
  epmautomate setDailyMaintenanceStartTime 14:20
  ```
- 将日常维护开始时间设置为太平洋标准时间 2:35 PM：
  ```
  epmautomate setDailyMaintenanceStartTime "14:35 America/Los Angeles"
  ```

setDemoDates

根据需要更新 Oracle 内部演示数据。

此命令仅在安装设置时用于 Oracle 内部演示数据。

仅适用于 Account Reconciliation：此命令还会为关联 `Demo Code` 属性值等于 `setdemodates` 或 `setdemodatesnostatuschange` 的所有调节重置日期。此命令最多处理 12 个期间的调节；即当前期间和前 11 个（历史）期间。如果两个以上期间的调节标记了 `Demo Code` 属性，则此命令将这些期间视为在前一期间内。没有此属性值的调节则不受影响。

- 如果值为 `setdemodates`，则此命令将根据指定的日期重置调节日期，并设置随机状态
- 如果值为 `setdemodatesnostatuschange`，则此命令将根据指定的日期重置调节日期而不更改调节状态

所有其他业务流程：此命令将重置任务开始日期和结束日期以及其他相关的日期信息，以使任务看起来适合演示。在计算新的任务日期时，将使用在任务的计划中设置的 `SETDEMODATES` 属性值以及您提供的 `Demo Date` 值。如果未指定 `Demo Date` 值，则此命令将使用当前的日期来计算新的任务日期。

> **注：**
>
> 调度中没有 `SETDEMODATES` 值的任务不受影响。

根据您指定的 `Demo Date`，此命令将与任务关联的所有日期向前推。其中包括核心运行时日期（开始日期、结束日期等等）和辅助日期（包括历史日期、各个工作流到期日和开始日期（实际））。任务状态不受影响。

适用于

Account Reconciliation、Enterprise Profitability and Cost Management、Financial Consolidation and Close、自由形式、Planning、Planning 模块、销售规划、战略性人员规划和 Tax Reporting

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate setDemoDates [DEMO_DATE]`，其中 *`DEMO_DATE`* 是格式为 `YYYY-MM-DD` 的可选日期。如果不指定此值，调节将被重置为当前日期。

示例

```
epmautomate setDemoDates 2020-02-15
```

setEJJournalStatus

在 Financial Consolidation and Close 中，设置 ERP 系统中的企业日记帐推送结果。可以使用此命令更新处于 `Post in Progress`（正在推送）状态的日记帐的推送状态，而不管其工作流状态如何。

此命令使用 CSV 文件来标识导入 ERP 系统的状态。可使用 uploadFile 命令将导入文件上传到环境。CSV 文件格式如下：

```
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"
```

“消息”列为可选，可以忽略。

此命令不会从 Financial Consolidation and Close 导出企业日记帐数据，也不会将其导入到 ERP 系统。

适用于

Financial Consolidation and Close

所需角色

服务管理员

用法

`epmautomate setEJJournalStatus` *`FILE_NAME`*`.csv [OPERATION=posting|validation]`，其中：

- *`FILE_NAME`* 标识包含导入到 ERP 系统的状态的 CSV 文件。
- `Operation`（可选）是要执行的操作。可能的值为 `posting` 和 `validation`。默认值为 `posting`

  `operation=posting` 提供一个包含日记帐推送结果的 CSV 文件，其中包含以下列：年、期间、日记帐 ID、推送状态（已推送/失败）、消息、错误代码和错误消息。“消息”、“错误代码”和“错误消息”列为可选列。日记帐的推送状态根据文件中提供的状态进行更新。仅更新推送状态为正在推送的日记帐。

  `operation = validation` 生成一个包含日记帐验证结果的 CSV 文件，其中包含以下列：年、期间、日记帐 ID、验证状态（有效/失败）、消息、错误代码、错误消息。“消息”、“错误代码”和“错误消息”列为可选列。日记帐的验证状态根据文件中提供的状态进行更新。仅更新验证状态为正在验证的日记帐。

示例

- 从文件设置企业日记帐推送结果：

  ```
  epmautomate setEJJournalStatus JournalStatus.csv
  ```

- 从文件设置企业日记帐推送结果：并执行验证操作：

  ```
  epmautomate esetEJJournalStatus JournalStatus.csv Operation=validation
  ```

setEncryptionKey

为数据库访问设置自定义加密密钥。

使用此命令可为客户提供自带密钥 (Bring Your own Key, BYOK) 解决方案，以在其标准密钥管理轮换中包括 Oracle Fusion Cloud Enterprise Performance Management。

自定义加密密钥在下一次日常维护环境之后开始生效。可以通过运行 resetService 命令立即将其激活。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate setEncryptionKey key=`*`key`*，其中 *`key`* 是要用作加密密钥的任意长度的自定义字符串。

示例

- 设置加密密钥：`epmautomate setEncryptionKey key=se!m+a2J`
- 删除加密密钥：`epmautomate setEncryptionKey key=`

setEssbaseQryGovExecTime

设置 Oracle Essbase 查询在终止之前可用于检索和传送信息的最长时间（以秒为单位）。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Profitability and Cost Management、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmautomate setEssbaseQryGovExecTime` *`TIME`*，其中 `TIME` 标识在多少秒之后终止 Essbase 查询。此值必须是不超过 70000 的整数。

Oracle 建议您不要将此值设置为 0（零），以防 Essbase 无限期运行。

示例

```
epmautomate setEssbaseQryGovExecTime 600
```

setIdleSessionTimeout

更改 Oracle Fusion Cloud Enterprise Performance Management 环境的会话超时（以分钟为单位）。新的会话超时将在下一次日常维护环境之后生效。可使用此命令将默认会话超时（75 分钟）更改为其他值。在会话空闲了使用此命令指定的持续时间后，用户将被重定向到登录页。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate setIdleSessionTimeout` *`MINUTES`*，其中 MINUTES 标识新的空闲会话超时（以分钟为单位，最短 15 分钟，最长 150 分钟）。

示例

```
epmautomate setIdleSessionTimeout 30
```

setIPAllowlist

配置允许访问 Oracle Fusion Cloud Enterprise Performance Management 的 IP 地址和无类别域间路由 (Classless Inter-Domain Routing, CIDR) 的允许列表。此命令添加或删除 IPv4 地址和 CIDR。

此命令提供了一种自助服务方法来为云 EPM 环境配置允许列表。

> **Note:**
>
> - 为云 EPM 环境设置 IP 允许列表时，只允许从这些特定 IP 地址进行连接。在这种情况下，除非将请求环境所在的数据中心或区域的出站 IP 地址添加到 IP 允许列表中，否则从另一个云 EPM 请求访问将不起作用。请参阅《运维指南》中的“云 EPM 数据中心和区域的出站 IP 地址”，了解您需要添加的 IP 地址，以确保其他环境可以与正在设置 IP 允许列表的环境进行通信。
> - 如果您在环境所在的身份云服务中使用 IDCS 网络边界，则您也必须在此网络边界中添加相同的 IP 地址。您可以选择仅在网络边界中添加 IP 地址，而不是为单个环境设置允许列表。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmAutomate setIPAllowlist add|remove` *`FILE_NAME`*`.txt`，其中：

- `add` 将文本文件中列出的 IP 地址和 CIDR 添加到允许列表。
- `remove` 从允许列表中删除文本文件中列出的 IP 地址和 CIDR。
- *`FILE_NAME`* 是文本文件的名称，该文件中列出要在允许列表中添加或删除的 IP 地址和 CIDR。文件中的每个条目必须用换行符分隔。使用 uploadFile 命令将此文件上传到环境。文件中的每一行都必须是 IPv4 地址或 CIDR，格式如下：

```
xxx.xxx.xxx.xxx
xxx.xxx.xxx.xxx/n
```

> **Note:**
>
> – 仅支持 IPv4 IP 地址。
> – 要指定连续的 IP 地址范围，应使用 CIDR 格式，而不是单个的 IP 地址。
> – 要禁用允许列表以允许来自任何 IP 地址的访问，请使用 getIPAllowlist 命令将所有现有 IP 地址和 CIDR 写入到文件。将文件上传到环境，然后使用 `remove` 选项运行此命令，如下例所示：
>
> ```
> epmAutomate getIPAllowlist > myRemoveList.txt
> epmAutomate uploadFile myRemoveList.txt
> epmAutomate setIPAllowlist remove myRemoveList.txt
> ```

示例

- 将一些 IP 地址和 CIDR 添加到允许列表：

```
epmAutomate setIPAllowlist add myAddList.txt
```

- 从允许列表中删除一些 IP 地址：

```
epmAutomate setIPAllowlist remove myRemoveList1.txt
```

setManualDataAccess

指定在紧急情况下是否允许 Oracle 手动访问环境的关系数据库和 Oracle Essbase 数据库，即环境无响应且客户尚未提交调查环境并使环境可用的服务请求。

在紧急情况下，Oracle 采用一种内部流程，即高级开发主管在执行独立的验证流程之后，允许 Oracle 手动访问关系数据库和 Essbase 数据库。您可以使用此命令禁止 Oracle 在未经您明确批准的情况下访问这些数据库。另外，您可以选择禁止 Oracle 在紧急情况下手动访问关系数据库和 Essbase 数据库，即使提交了服务请求也是如此。

使用此命令指定的设置将立即生效。默认情况下，允许 Oracle 进行手动数据访问。使用此命令撤消此访问权限。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost

Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate setManualDataAccess Allow|Revoke [disableEmergencyAccess=true|false]`，其中 `disableEmergencyAccess`（可选）指定是否要禁止对关系数据库和 Essbase 数据库的所有手动访问。将此属性值设置为 `true` 会禁止 Oracle 手动访问这些数据库，即使提交了服务请求也是如此。默认值为 `false`。
如果将 `disableEmergencyAccess` 的值指定为 `true`，那么在诊断故障和修复停止运行的环境时，Oracle 会无法访问关系数据库和 Essbase 数据库。如果环境停止运行，您将无法发出此命令来允许 Oracle 手动访问这些数据库。

示例

- 撤消在紧急情况下未经明确批准手动访问关系数据库和 Essbase 数据库的权限：
  `epmautomate setManualDataAccess revoke`
- 允许在紧急情况下手动访问关系数据库和 Essbase 数据库：
  `epmautomate setManualDataAccess allow`
- 禁止手动访问关系数据库和 Essbase 数据库，即使提交了服务请求也是如此：
  `epmautomate setManualDataAccess revoke disableEmergencyAccess=true`

setPeriodStatus

为期间设置特定状态。

适用于

Account Reconciliation

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

具有超级用户、用户和查看者预定义角色的用户可能需要其他应用程序角色。

用法

`epmautomate setPeriodStatus` *`PERIOD STATUS`*，其中：

- *`PERIOD`* 是期间名称
- *`STATUS`* 是 `OPEN`、`CLOSED` 或 `LOCKED`

示例

```
epmautomate setPeriodStatus "January 2015" OPEN
```

setRestrictedDataAccess

配置 Oracle Fusion Cloud Enterprise Performance Management 环境以阻止服务管理员在使用“提供反馈”实用程序时同意向 Oracle 提交应用程序快照。

如果您有不允许 Oracle 访问您的数据的策略，请在向 Oracle 提交反馈时使用此命令管理数据共享。为您的环境设置受限数据访问将禁用“提供反馈”实用程序显示的“提交应用程序快照”选项。

适用于

Account Reconciliation、Oracle Fusion Cloud Enterprise Data Management、Enterprise Profitability and Cost Management、Financial Consolidation and Close、自由形式、Planning、Planning 模块、Profitability and Cost Management、Narrative Reporting、销售规划、战略性人员规划和 Tax Reporting。

所需角色

服务管理员

用法

`epmautomate setRestrictedDataAccess true|false`
要查看当前数据访问限制状态，请使用 getRestrictedDataAccess。

示例

- 阻止服务管理员同意提交应用程序快照：
  `epmautomate setRestrictedDataAccess true`
- 允许服务管理员同意提交应用程序快照：
  `epmautomate setRestrictedDataAccess false`

setSubstVars

在应用程序或多维数据集级别创建或更新替代变量。

不能使用该命令为替代变量设置多个值和/或函数。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate setSubstVars` *`CUBE_NAME SUBSTVAR=VALUE [SUBSTVAR=VALUE]`*，其中：

- *`CUBE_NAME`* 是为其创建或更新替代变量的多维数据集（例如，Plan1、Plan2）。使用 `All` 代替多维数据集名称可以在应用程序级别设置或更新替代变量。
- *`SUBSTVAR`* 是要为其设置或更新值的替代变量的名称。
- *`VALUE`* 是替代变量的新值。

示例

- 在应用程序级别创建或更新一个替代变量：`epmautomate setSubstVars ALL CurYear=2015 CurPeriod=Jan`
- 在多维数据集级别创建或更新多个替代变量：`epmautomate setSubstVars Plan2 CurYear=2013 CurPeriod=Jan`

setVirusScanOnFileUploads

允许 OCI（第 2 代）环境在文件上传到 Oracle Fusion Cloud Enterprise Performance Management 之前扫描文件中是否有病毒。

所有 OCI（第 2 代）环境都使用防病毒程序进行保护。此命令允许您对上传文件启用病毒扫描来增强安全性。在上传文件之前扫描文件可防止将病毒上传到环境。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate setVirusScanOnFileUploads true|false`
默认情况下，病毒扫描处于禁用状态（设置为 `false`）。当此值设置为 `true` 时，云 EPM 扫描所有上传文件。如果文件感染了病毒，则不会将文件上传到环境。

示例

- 启用病毒扫描：`epmautomate setVirusScanOnFileUploads true`
- 禁用病毒扫描：`epmautomate setVirusScanOnFileUploads false`

simulateConcurrentUsage

通过模拟用户在环境中执行不同的并发操作。

可以使用此命令验证环境的性能，以确认在特定数量的用户运行特定操作期间，服务承受负载时响应时间是否可接受。例如，可以使用此命令衡量 50 个用户同时使用不同的 POV 打开某个表单时的性能。还可以使用此命令自助对环境进行负载测试。

此命令通过按给定的用户数和迭代次数执行指定操作来执行模拟。它将运行多次迭代以计算特定操作的最短时间、最长时间和平均时间。它支持以下操作以便执行并发使用负载测试：

- 打开表单
- 保存表单
- 运行业务规则
- 运行业务规则集
- 运行数据规则

- 打开即席网格
- 执行报表
- 执行工作簿

> **Note:**
>
> 此命令不支持 Financial Reporting 报表和工作簿；仅支持属于“报表”（以前称为“管理报表”）的工作簿和报表。

> **Caution:**
>
> 此命令在当前环境中执行指定的操作，并可能根据操作更新环境中的数据。在测试环境中运行此命令。不建议在生产环境中运行此命令。

此命令接受 ZIP 文件（必须已经上传到环境的收件箱）作为输入。该 ZIP 文件包含一个 `requirement.csv` 文件和输入文件（用于支持 `requirement.csv` 中包含的用例）。（可选）ZIP 文件可以包含 `userVarMemberMapping.csv` 文件（用于提供用户变量成员映射）、`options.xml` 文件（用于为某些用例提供 Oracle Smart View for Office 选项），以及 `users.csv` 文件（用于提供现有用户的用户名和密码以供使用，而不是创建新用户）。之后，该命令模拟用例并创建报表，该报表可通过电子邮件发送给一个或多个收件人。

> **Note:**
>
> 此命令不生成“提供反馈”提交。您可以使用业务流程屏幕中的“提供反馈”（请参阅《*Oracle Enterprise Performance Management Cloud* 管理员入门》中的“使用提供反馈实用程序帮助 Oracle 收集诊断信息”）选项、提供反馈 REST API 或 feedback 命令来生成“提供反馈”提交，以便在运行模拟后获取环境的详细信息。

使用方案 1：对 50 个用户同时打开表单时的应用程序性能进行接受度测试。

解决方案：

1. 创建 `requirement.csv`，其中包含类似以下内容的条目，假设您要打开存储在 `Library/Global Assumption/` 文件夹中的名为 `Exchange Rates` 的表单：

   ```
   # 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. 使用“打开表单输入文件”中指定的格式创建 `open_form_input.csv`。此文件中将有一个条目，将使用此条目 50 次。如果要使用不同的 POV 打开同一表单，则具有的条目数将与要使用的 POV 数相同。
3. 如果需要设置用户变量成员映射，请使用“创建 UserVarMemberMapping.csv 文件”中指定的格式创建 `userVarMemberMapping.csv`。
4. 如果需要使用 Smart View 选项，则将 Smart View 选项导出到 `options.xml` 中。有关详细信息，请参阅“创建 options.xml 文件”。
5. 创建包含前面步骤中的文件的 ZIP 文件，并将其上传到收件箱。

6. 使用上一步中的 ZIP 文件作为输入文件来运行 `simulateConcurrentUsage` 命令。

使用方案 2：模拟特定时间（例如财务年度结束时）季节性使用量增加时的性能。假设：100 个用户保存某个表单，各个用户之间存在六秒延时时间。

解决方案：

1. 创建 `requirement.csv`，其中包含类似以下内容的条目，假设您要保存存储在 `Library/Dashboards/` 文件夹中的名为 `Accessories Revenue` 的表单：

   ```
   # Type of Operation,Artifact Name,Number of Users,Input File,Additional Info
   Save Form, Library/Dashboards/Accessories Revenue,100,save_form_input.csv,
   ```

2. 使用“保存表单输入文件”中指定的格式创建 `save_form_input.csv`。
3. 如果需要设置用户变量成员映射，请使用“创建 UserVarMemberMapping.csv 文件”中指定的格式创建 `userVarMemberMapping.csv`。
4. 如果需要使用 Smart View 选项，则将 Smart View 选项导出到 `options.xml` 中。有关详细信息，请参阅“创建 options.xml 文件”。
5. 创建包含前面步骤中的文件的 ZIP 文件，并将其上传到收件箱。
6. 使用上一步中的 ZIP 文件作为输入文件以及属性值 `iteration=1` 和 `lagTime=6` 来运行 `simulateConcurrentUsage` 命令。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、战略性人员规划和销售规划。

所需角色

服务管理员。您还需要身份域管理员角色来使用 testModes 0、1 和 2。

用法

`epmautomate simulateConcurrentUsage` *`INPUT_FILE.zip`* `[iterations=`*`COUNT`*`] [notificationEmails="EMAIL_ADDRESS"] [testMode=0|1|2|3|4] [lagTime=`*`LAG_TIME`*`]`，其中：

- *`INPUT_FILE.zip`* 是标识用例的 ZIP 文件的名称。在运行此命令之前，先使用 uploadFile 命令（示例命令语法 `epmautomate uploadFile "C:/uploads/`*`INPUT_FILE.zip`*`" inbox`）将此文件上传到收件箱。此 ZIP 文件必须包含以下文件：
  - 一个名为 `requirement.csv` 的用例 CSV 文件。此 CSV 文件中的每一行标识要执行的操作类型、对象名称、并发用户数、指定操作详细信息的输入文件以及与每个用例相关的其他信息。请参阅“创建 requirement.csv 文件”。
  - 包含每个操作详细信息的输入文件。请参阅以下主题：
    * 打开表单输入文件
    * 保存表单输入文件
    * 运行业务规则输入文件
    * 运行业务规则集输入文件
    * 运行数据规则输入文件
    * 即席网格输入文件

    - 执行工作簿输入文件
    - 执行报表输入文件
  - （可选）将 `userVarMemberMapping.csv` 添加到输入 ZIP 文件中以提供用户变量成员映射。请参阅“创建 UserVarMemberMapping.csv 文件”。
  - （可选）将 `options.xml` 添加到输入 ZIP 文件中以使用 Smart View 选项。请参阅“创建 options.xml 文件”。
  - （可选）将 `users.csv` 添加到输入 ZIP 文件中以提供现有用户的用户名和密码。请参阅“创建 users.csv 文件”。
- `iterations` 是一个正数，指示运行 `requirement.csv` 中标识的每个用例以衡量响应时间的次数。如果未指定，则操作仅运行一次。
- `notificationEmails`（可选）指示要向其发送此命令结果的电子邮件地址。如果指定多个电子邮件地址，则使用分号分隔这些地址。此外，还使用双引号括住地址列表。如果未指定，则将向发出命令的用户发送结果。有关此报表的详细信息，请参阅“"Simulate Concurrent Usage Report"（模拟并发使用报表）示例”。
- `[testMode]`（可选）指定并发使用模拟模式。默认值为 0。可接受的值包括：
  - `0`：默认模拟模式，在此模式下，将模拟用户添加到环境，并为其分配服务管理员角色，运行模拟，然后删除模拟用户。如果要仅运行一次测试，则此模式适用。
    模拟用户具有以下属性：

    名字：testuser1、testuser2 等

    姓氏：testuser1、testuser2 等.

    电子邮件地址：testuser1@discard.oracle.com、testuser2@discard.oracle.com 等

    用户名：testuser1、testuser2 等
  - `1`：将模拟用户添加到环境，并为其分配服务管理员角色。不运行模拟，也不删除模拟用户。
    使用此模式后，使用模式 3 运行该命令以运行所需次数的模拟。最后，使用模式 2 运行该命令以删除模拟用户。
  - `2`：删除模拟用户。不创建用户，也不运行模拟。
  - `3`：使用已有的模拟用户运行模拟，而不添加或删除用户。
  - `4`：使用输入 ZIP 文件中包含的 `users.csv` 文件中定义的用户来运行该命令。请参阅“创建 users.csv 文件”。此模式不会为模拟创建用户。而是，使用现有用户。

  如果要仅运行一次并发使用，则使用 `testMode=0`。要运行一系列测试：
  - 首先，使用 `testMode=1` 运行该命令以添加模拟用户并为其分配服务管理员角色。
  - 然后，使用 `testMode=3` 运行该命令以运行所需次数的模拟。
  - 最后，使用 `testMode=2` 运行该命令以删除模拟用户。
- `[lagTime]`（可选）指定命令在执行 `requirement.csv` 中的每个用例之间应等待的秒数（5 秒或更长）。默认值为 5 秒。请勿使用负数（例如 -1）、分数（例如 1/2）和小数值。
  在启动一个用户执行 `requirement.csv` 中的一个用例后，该命令等待此参数指定的秒数，然后启动下一个用户执行该用例。由于用户活动通常不是同时启动的，因此设置此参数有助于在环境中创建更真实的负载模拟。

示例

```
epmautomate simulateConcurrentUsage test_simulation.zip iterations=5
notificationEmails="jane.doe@example.com;john.doe@example.com;example@example.
com" lagTime=6
```

skipUpdate

要求 Oracle 跳过对环境应用每月更新（最多连续三个周期），或者删除以前使用此命令发出的所有跳过更新请求，以便将环境更新到主代码行。

您还可以使用此命令列出当前为某个环境指定的跳过更新请求。使用此命令跳过环境更新后生成的活动报表中包含环境的跳过更新状态（位于运行量度中）。请参阅《管理员入门指南》中的“运行量度”。

将继续向环境应用当月的每周和紧急修补程序（如果有）。对请求了延迟升级的月份将不进行更新。

对于应用了一次性修补程序的环境，不能跳过更新。此外，如果每月更新距离环境当前采用的更新超过三个月，则不能跳过该每月更新。例如，如果环境当前采用 23.12，则可以跳过 24.01、24.02 和 24.03，但不能跳过 24.04。有关更新延迟工作原理的详细信息，请参阅《运维指南》中的“请求延迟升级生产环境”。

> **注：**
>
> 如果您只对其中一个环境跳过三个月的更新（例如，跳过对生产环境的更新，而不跳过对测试环境的更新），则这些环境将相差三个版本。在这种情况下，您可能无法在这些环境间迁移快照。
> 例如，假设您的测试环境和生产环境当前采用 23.12，并且您只对生产环境跳过 24.01、24.02 和 24.03 版更新。当版本 24.03 变为可用时，您的测试环境将采用版本 24.03，而生产环境仍采用版本 23.12。在这种情况下，不支持在测试环境和生产环境之间进行迁移。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate skipUpdate add|remove|list [version=`*`UPDATE_NUMBER`*
`comment="`*`COMMENT`*`"]`，其中：

- `add` 为特定的每月更新设置跳过更新请求。必须指定以下参数：
  - `version`：应跳过的每月更新。可以跳过即将到来的三个每月更新中的一个、两个或三个。例如，如果环境采用 23.12 每月更新，可以跳过 24.01、24.02、24.03 或其中任何更新。要跳过三个每月更新，则运行三次该命令，每次使用版本参数指定要跳过的一个

特定更新，例如依次分别指定 `version=24.01`、`version=24.02` 和 `version=24.03`。在这种情况下，环境将更新到 24.04 每月周期中的主代码行。

如果请求跳过更新的每月周期与当前每月周期之间存在差距，Oracle 将根据需要更新环境，然后跳过指定的每月周期中的更新。例如，环境采用 23.12 每月更新，并且您指定跳过版本 24.02 和 24.03 的更新。在这种情况下，将在 24.01 中更新环境；将跳过 24.02 和 24.03 更新。环境将更新到 24.04 中的主代码行。

- `comment`：说明为何需要跳过更新的文本。注释必须用双引号括起来。

- `remove` 删除为环境指定的所有跳过更新请求，以便可在下一个日常维护期间将其更新到主代码行。如果您对某个环境有多个跳过更新请求，此命令将删除所有这些请求。
- `list` 显示当前为环境设置的跳过更新请求（发出跳过更新请求的用户的登录 ID、注释、要跳过更新的版本和发出请求的日期），如下图所示：

```
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
```

示例

- 请求跳过更新：`epmautomate skipUpdate add version=24.01 comment="We are in the process of closing the quarter"`
- 查看跳过更新详细信息：`epmautomate skipUpdate list`
- 删除所有跳过更新请求：`epmautomate skipUpdate remove`

snapshotCompareReport

比较两个快照，并创建“快照比较报表”来标识快照包含的计算规则与规则集以及数据表单中的差异。

该报表显示规则和表单定义中的差异。它还标识一个快照是否包含另一个快照中的规则、规则集或表单。规则集中的差异根据其中包含的规则进行标识。请注意，生成的报表不报告多维数据集或维属性中的差异。

- 环境中最近发生性能下降。您可以将以前的快照与当前快照进行比较，以检查可能导致性能下降的差异。
- 您预计两个环境应具有相同的功能行为或性能，您想要查看这两个环境在行为或性能上的差异。在这种情况下，您可以比较两个环境的快照，以了解两者的差异。
- 您怀疑环境中的某些规则或表单消失了。使用此报表可以比较过去存在的对象与当前对象。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、战略性人员规划和销售规划。

所需角色

服务管理员

用法

`epmAutomate snapshotCompareReport` *`SOURCE_SNAPSHOT TARGET_SNAPSHOT`* `[reportName=`*`REPORT_NAME`*`.html]`，其中：

- *`SOURCE_SNAPSHOT`* 是比较所基于的快照的名称。报表包含了有关此快照中规则和表单差异的数据。
- *`TARGET_SNAPSHOT`* 是要比较的快照的名称。

> **Note:**
>
> – 可以指定带有或不带 .ZIP 扩展名的快照名称。
>
> – 两个快照都必须在环境中可用。可使用 uploadFile、copyFromObjectStorage 或 copySnapshotFromInstance 命令将快照上传到环境中。

- *`REPORT_NAME`*（可选）是报表文件的名称。默认报表名称为 `SnapshotCompare.html`。使用 downloadFile 命令下载报表。

示例

- `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

对“实体”、“帐户”、“方案”、“版本”及自定义维的成员进行排序。

在将成员加载到应用程序后对其进行排序时，此命令非常有用。

> **注：**
>
> 不能使用此命令对“期间”、“年”或“货币”维成员进行排序。

适用于

Planning、Planning 模块、自由形式、Enterprise Profitability and Cost Management、销售规划和战略性人员规划。

所需角色

服务管理员

用法

`epmautomate sortMember Member [type=children|descendants] [order=ascending|descending]`，其中：

- *Member* 是要对其后代或子代进行排序的父代成员的名称。
- `type` 为可选，用于指定要排序的成员。可接受的值包括：
  - `descendants` 可对您指定为 `Member` 值的父代成员的所有子成员（子代和后代）进行排序
  - `children` 是默认值，仅对您指定为 `Member` 值的父代成员的下一级成员进行排序。
- `order` 为可选，用于指定排序顺序。可接受的值包括：
  - `ascending`；这是默认排序顺序。
  - `descending`

示例

- 按升序对“实体”维的子代进行排序：`epmautomate sortMember Entity`
- 按降序对“实体”维的所有子成员进行排序：`epmautomate sortMember Entity type=descendants order=descending`

unassignRole

对于此命令中使用的 ANSI 或 UTF-8 编码 CSV 文件中包含其登录 ID 的用户，删除当前分配给这些用户（包括运行此命令的用户）的角色。您可以使用此命令删除预定义角色或应用程序角色的分配。

CSV 文件格式如下：

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

服务管理员使用 uploadFile 命令将文件上传到环境中。

> **注：**
>
> 使用双引号将包含空白字符的角色名称括起来。

完成时，该命令会将有关每个失败条目的信息输出到控制台上。查看此信息可了解对 CSV 文件中的某些条目执行命令失败的原因。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost

Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 要删除预定义角色分配：
  - 服务管理员和/或任何预定义角色
  - 身份域管理员和任何预定义角色
- 要删除应用程序角色：
  - 服务管理员
  - 任何预定义角色和访问控制 - 管理应用程序角色

用法

`epmautomate unassignRole` *`FILE_NAME ROLE`*，其中：

- *`FILE_NAME`* 是 CSV 文件的名称，该文件中包含要撤消其角色分配的用户的登录 ID。以小写形式指定 CSV 扩展名。
  用户登录值不区分大小写。例如，`jane.doe@example.com` 被视为与 `Jane.Doe@Example.com` 或其任何大小写变体相同。
- `ROLE` 标识以下角色之一。角色名称不区分大小写。
  - 若要删除用户的预定义角色分配，则 `ROLE` 应标识适用于服务的预定义角色。请参阅《管理员入门指南》中的“了解预定义角色”。
  - 如果您删除用户的应用程序角色分配，`ROLE` 应标识属于当前环境中应用程序的角色。访问控制的**角色**选项卡中列出了应用程序角色。有关每个业务流程的应用程序角色说明，请参阅《管理 *Oracle Enterprise Performance Management Cloud* 访问控制》中的以下主题：
    - Account Reconciliation
    - Enterprise Profitability and Cost Management
    - Planning、自由形式、Financial Consolidation and Close 和 Tax Reporting
    - Profitability and Cost Management
    - Oracle Enterprise Data Management
    - Narrative Reporting

示例

- 取消分配用户的预定义身份域角色：
  ```
  epmautomate unassignRole remove_roles.csv "Service Administrator"
  ```
- 取消分配用户的应用程序角色：
  ```
  epmautomate unassignRole example_file.csv "Task List Access Manager"
  ```

updateGuidedLearningSettings

设置在环境中启用 Oracle Guided Learning (OGL) 所需的值。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Enterprise Profitability and Cost Management、销售规划和战略性人员规划

所需角色

服务管理员

用法

`epmautomate updateGuidedLearningSettings [oglAppId=APP_ID] [oglServerUrl=OGL_SERVER_URL]`，其中：

- `OglAppId`（可选）是 OGL 应用程序 ID。这是 OGL 服务器上指南（内容）的逻辑分组的 ID。此值最多可包含 100 个字符。
- `OglServerUrl`（可选）是您拥有活动 OGL 帐户的 OGL 服务器的 URL。默认值为 `https://guidedlearning.oracle.com`。此 URL 必须指定协议，并且最多可以有 300 个字符。

> **Note:**
>
> 至少需要上述参数之一。

示例

```
epmautomate updateGuidedLearnignSettings "oglAppId=123xyz" "oglServerUrl=https://guidedlearning.oracle.com"
```

updateUsers

使用新值修改身份域中的 Oracle Fusion Cloud Enterprise Performance Management 用户属性（例如电子邮件、名字和姓氏），这些值在上传到环境的 ANSI 或 UTF-8 编码的逗号分隔值 (CSV) 文件中标识。

服务管理员使用 uploadFile 命令将文件上传到环境中。CSV 文件中的所有列都是必填的；您必须在每列中提供有效的条目。此命令验证这些必填值的每个定义，并显示标识每个缺少或无效值的错误消息。输入文件格式如下：

```
First Name,Last Name,Email,User Login
Jane,Doe,jane.doe@example.com,jdoe
John,Doe,john.doe@example.com,john.doe@example.com
```

如果 CSV 文件中的用户登录值与身份域中存在的帐户匹配，则该命令会修改用户帐户以匹配输入文件中的值。由于用户帐户在身份域支持的所有环境中是公用的，因此更新的用户信息会提供给共享身份域的所有环境。分配给用户的预定义角色和特定于应用程序的角色不受此命令的影响

> **Note:**
>
> - 不能使用此命令修改用户登录值。
> - 不允许您修改自己的帐户属性。
> - 包含多字节字符的输入文件必须使用 UTF-8 字符编码。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、战略性人员规划和销售规划。

所需角色

身份域管理员和任何预定义角色

用法

`epmautomate updateUsers` *`FILE_NAME`*，其中 *`FILE_NAME`* 是包含要修改的用户信息的 CSV 文件名称。

示例

```
epmautomate updateUsers update_user_info.csv
```

upgrade

自动下载并静默安装 EPM Automate 的最新版本。

运行 login 命令启动会话后，EPM Automate 会识别当前安装的版本。如果安装的版本不是最新的可用版本，它会通知您有较新版本可用。

> **注：**
>
> 仅当登录的用户为 Windows 管理员时，才能升级 Windows 管理员部署的 EPM Automate。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户
- 用户
- 查看者

用法

```
epmautomate upgrade
```

示例

```
epmautomate upgrade
```

uploadFile

将文件从本地计算机上传到服务。使用此命令可以上传包含数据、元数据、规则定义、维定义、映射的事务、模板和备份快照的文件。

此命令不覆盖环境中的现有文件。如果在上传的文件与上传位置中的某个文件同名，它会显示错误。

此命令不允许您上传二进制文件（例如，扩展名为 .exe、.com、.cmd、.bin、.bat、.msi 和 .vbs 等的文件）以及具有禁止扩展名的文件。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 超级用户和“迁移 - 管理”应用程序角色（Oracle Enterprise Data Management Cloud 和 Profitability and Cost Management）

用法

`epmautomate uploadFile "FILE_NAME" [UPLOAD_LOCATION]`，其中：

- *FILE_NAME* 为文件的名称，包括绝对路径（如果文件不在您从中运行 EPM Automate 的目录中）。
- *UPLOAD_LOCATION*（可选）为要将文件上传到的 Oracle Fusion Cloud Enterprise Performance Management 位置。如果要将文件上传到默认上传位置，请不要指定上传位置。有关详细信息，请参阅“默认文件位置”。支持的值包括：
  - `inbox`，将文件上传到收件箱中。除 Profitability and Cost Management 之外，云 EPM 业务流程都在此位置查找要处理的文件。
  - `profitinbox`，上传要由 Profitability and Cost Management 处理的文件。
  - `to_be_imported`，上传要在环境的下一次日常维护期间导入的 Narrative Reporting 快照。
  - `inbox/directory_name`，将文件上传到收件箱中的目录，以供数据管理处理。
  - `outbox`，将文件上传到 Profitability and Cost Management 之外的业务流程使用的发件箱。
  - `profitoutbox`，将文件上传到 Profitability and Cost Management 使用的发件箱。

示例

- 将快照上传到默认位置：
  ```
  epmautomate uploadFile "C:/snapshots/backup_snapshot.zip"
  ```
- 将文件上传到数据管理收件箱：
  ```
  epmautomate uploadFile "C:/pbcsdata/quarterlydata.csv" inbox
  ```
- 将文件上传到收件箱中的文件夹（用于数据管理）：
  ```
  epmautomate uploadFile "C:/fdmee_data/data.zip" inbox/dm_folder
  ```

- 将文件上传到 profitinbox (Profitability and Cost Management)：
  `epmautomate uploadFile "C:/profitability_data/data.zip" profitinbox`
- 将 Narrative Reporting 快照从 `C:\temp` 目录上传到 to_be_imported 位置：
  `epmautomate uploadFile "C:\temp\EPRCS_Backup.tar.gz" to_be_imported`

userAuditReport

生成用户审核报表（.CSV 文件）并将其存储在默认下载位置中。

用户审核报表包含在指定的一段时间（最多过去 120 天）内登录到环境的用户的相关信息。它会列出用户登录 ID、用户从中登录的计算机的 IP 地址以及用户访问环境的日期和时间（例如 July 28, 2022 18:43:21 UTC）。

> **注：**
>
> 对于在 5 分钟内多次登录 Oracle Fusion Cloud Enterprise Performance Management 环境的用户，用户审核报表仅列出一个登录条目。

	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

使用 downloadFile 将生成的报表下载到您的计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmautomate userAuditReport` *`FROM_DATE TO_DATE REPORT_NAME`*，其中：

- *`FROM_DATE`* 指明要生成审核报表的期间开始日期（格式为 `YYYY-MM-DD`）
- *`TO_DATE`* 指明要生成审核报表的期间结束日期（格式为 `YYYY-MM-DD`）
- *`REPORT_NAME`* 是报表文件的名称

> **注：**
>
> 只能为过去 120 天生成此报表。

示例

```
epmautomate userAuditReport 2016-10-15 2016-12-15 myAuditReport.CSV
```

userGroupReport

生成一个报表（CSV 文件），其中列出要在访问控制中向其分配用户的组，并将该报表存储在默认的下载位置。

该报表会指示向组中分配的用户是直接的（作为组成员）还是间接的（作为嵌套组子代的组成员）。

该报表会按以下格式标识用户的登录名、名字、姓氏、电子邮件地址、分配的组以及分配类型。此报表与从访问控制中的“用户组报表”选项卡创建的 CSV 版本报表相同。例如，假设 `jdoe` 是组 `Test1` 的成员，且该组是嵌套组 `Test2` 的子代。在此方案中，报表会显示 `jdoe` 的以下信息：

```
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
```

使用 downloadFile 将生成的报表下载到您的计算机。

适用于

Planning、Planning 模块、自由形式、Financial Consolidation and Close、Tax Reporting、Account Reconciliation、Profitability and Cost Management、Enterprise Profitability and Cost Management、Oracle Fusion Cloud Enterprise Data Management、Narrative Reporting、销售规划和战略性人员规划。

所需角色

- 服务管理员
- 任何预定义角色和访问控制 - 管理应用程序角色
- 任何预定义角色和“访问控制 - 查看”应用程序角色

用法

`epmautomate userGroupReport` *`REPORT_NAME`*，其中 *`REPORT_NAME`* 是报表文件的名称。

示例

```
epmautomate userGroupReport UsrGrpReport.CSV
```

validateConsolidationMetadata

验证环境的元数据以确保数据库刷新和合并不出错。

在使用 importMetadata 命令导入元数据后，使用此命令验证这些元数据以确保在运行 refreshCube 命令时数据库刷新不出错。如果合并元数据不正确，则合并可能也会失败。

运行此命令时，会在运行它的控制台上显示 0（零）个验证错误或验证错误数。验证错误会写入一个 CSV 文件，您可以使用此文件更正元数据错误。使用 downloadFile 命令将生成的 CSV 文件下载到本地服务器。

适用于

Financial Consolidation and Close

所需角色

服务管理员

用法

`epmautomate validateConsolidationMetadata LOG_FILE_NAME`，其中 `LOG_FILE_NAME` 标识文件的名称，此文件将包含有关此命令发现的错误的信息。

示例

```
epmautomate validateConsolidationMetadata validation_error.csv
```

validateModel

验证 Enterprise Profitability and Cost Management 模型并将验证输出写入到文件。

适用于

Enterprise Profitability and Cost Management

所需角色

服务管理员

用法

`epmautomate validateModel "modelName" FILE_NAME.txt [messageType=All|Warning|Error] [ruleStatus=All|Enabled|Disabled]`，其中：

- `modelName` 是要验证的 Enterprise Profitability and Cost Management 模型的名称。此值必须用双引号括起来。
- *`FILE_NAME`* 是该命令应向其写入模型验证输出的文本文件的唯一名称。此文件在发件箱中创建，可以使用 downloadFile 命令进行下载。
- `messageType`（可选）是要包括在模型验证输出中的信息的状态。可能的参数值为：
  - `All`，将错误和警告都写入验证输出文件中。
  - `Error`，仅将错误记录在验证输出文件中。这是默认值。
  - `Warning`，仅将模型验证警告记录在验证输出文件中。
- `ruleStatus`（可选）是以下值之一：
  - `All`，验证选定模型的所有规则。这是默认值。
  - `Enabled`，仅验证已启用的规则。
  - `Disabled`，仅验证已禁用的规则。

示例

```
epmautomate validateModel "10 Actuals Allocation Process" results.txt
messageType=All ruleStatus=Enabled
```

退出代码

EPM Automate 将返回一个退出代码和一条消息以指明操作的状态。退出代码按五个代码编号分组；每个代码可以指示许多错误情况。查看随附的消息可以确定导致错误的具体情况。

此外，EPM Automate 会为每次失败的命令执行创建日志文件（*COMMANDNAME_TIMESTAMP.log*，例如 `uploadfile_16_11_2016_11_27_10.log`）。将在运行 EPM Automate 的计算机上创建日志文件。

退出代码 1 错误

`Command failed to execute` EPM Automate 使用此退出代码显示与 HTTP 状态代码 200 和 400 相关的消息。EPM Automate 使用的 REST API 返回这些代码。

`Insufficient privileges to perform the operation` 如果登录到服务的用户所用的凭据没有足够权限来执行您尝试的操作，将显示此错误。

使用具有足够权限来执行该操作的帐户进行登录。通常，仅允许服务管理员在服务中执行操作。

`Resource does not exist` 如果服务中不存在您要删除或下载的文件或快照，将显示此错误。

使用 `listfiles` 命令验证文件名及其位置。

`Invalid snapshot SNAPSHOT` 服务无法验证您为导出或导入操作指定的快照时，将显示此错误。

验证您使用的快照是否有效。

`Internal server error.Unable to delete file: FILE_NAME Please issue "Provide Feedback" with details` 如果由于服务器错误而无法从服务中删除文件或快照，将显示此错误。

使用 Feedback 命令或“提供反馈”功能向 Oracle 报告此问题。

`Invalid file: FILE_NAME` 如果服务中不存在您要删除或下载的文件或快照，或者如果文件名不是所需格式，将显示此错误。

使用 `listfiles` 命令验证文件名及其位置。

`Recreate is running for a long time.Please contact support` 如果您启动的重新创建操作未在一个小时内完成，将显示此错误。

使用 feedback 命令或“提供反馈”功能向 Oracle 报告此问题。

`Reset service is running for a long time.Please contact support` 如果您启动的重置服务操作未在一个小时内完成，将显示此错误。

使用 feedback 命令或“提供反馈”功能向 Oracle 报告此问题。

`Cannot perform operation.Another instance is in progress.Please try after some time` 如果您在 `copysnapshotfrominstance` 命令的另一个实例处于活动状态时尝试执行该命令，将显示此错误。

等待 `copysnapshotfrominstance` 命令完成，然后再次尝试运行该命令。

Cannot perform operation.Another maintenance script is in progress.Please try after some time 如果您在日常维护或服务重置进程运行时尝试执行 `copysnapshotfrominstance`、`recreate` 或 `resetservice` 命令，将显示此错误。

等待维护或重置进程完成后重新运行该操作。

Login to source instance failed: *SOURCE_URL* 如果 EPM Automate 无法登录到源环境来启动 `copysnapshotfrominstance` 命令，将显示此错误

验证用于访问源环境的凭据、身份域和 URL 是否有效。

Internal server error.Copy snapshot from source instance failed.Please issue "Provide Feedback" with details 如果运行 `copysnapshotfrominstance` 命令时 EPM Automate 遇到意外问题，将显示此错误。

使用 `feedback` 命令或“提供反馈”功能向 Oracle 报告此问题。

Internal server error.Please issue "Provide Feedback" with details 有许多内部服务器问题会显示此错误，此错误表明需要 Oracle 采取更正措施。

使用 `feedback` 命令或“提供反馈”功能向 Oracle 报告此问题。

Snapshot *SNAPSHOT_NAME* already exists.Please delete the snapshot and rerun the command 将快照下载或上传到存在另一个同名快照的位置时，将显示此错误。

删除或移除现有快照，然后重试该命令。

Error in extracting the snapshot.Please retry with a proper snapshot 如果运行 `importsnapshot` 命令时 EPM Automate 无法提取快照内容，将显示此错误。

验证该快照是否有效。

Internal server error.Unable to open file for write.Please issue "Provide Feedback" with details 如果发生的错误（例如生成审核报表时出错）导致创建或更新 CSV 文件，将显示此错误。

使用 `feedback` 命令或“提供反馈”功能向 Oracle 报告此问题。

No matching records found, please select a different date range 如果您运行 `userauditreport` 命令来为没有审核数据可用的日期范围生成审核报表，将显示此错误。

指定有效的日期范围，然后重新运行 `userauditreport` 命令。请注意，服务仅维护过去 365 天的审核历史记录。

File with same name exists: *FILE_NAME*, please choose a different filename 如果服务中存在与您指定的审核报表名称相同的报表，将显示此错误。

从服务中删除现有文件或指定其他文件名，然后重新运行 `userauditreport` 命令。

Operation failed with status $1.Please issue "Provide Feedback" 此消息指出发生了内部服务器错误，该错误导致重置服务或重新创建服务进程失败。

使用 `feedback` 命令或“提供反馈”功能向 Oracle 报告此问题。

EPMCSS-20643: Failed to add users.File *FILE_NAME*.csv is not found.Please provide a valid file name 如果包含有关要添加的用户的信息的指定 CSV 文件在收件箱中不可用，将显示此错误。

使用 `listfiles` 命令验证文件名及其位置。如果该文件不在收件箱中，则使用 uploadFile 命令上传该文件。

EPMCSS-20644: Failed to remove users.File *FILE_NAME*.csv is not found.Please provide a valid file name 如果包含有关要删除的用户的信息的指定 CSV 文件在收件箱中不可用，将显示此错误。

使用 `listfiles` 命令验证文件名及其位置。如果该文件不在收件箱中，则使用 uploadFile 命令上传该文件。

20645: Failed to assign role for users.Invalid role name *role*.Please provide a valid role name 如果在 CSV 文件中指定的角色不受支持，将显示此错误。

验证文件中指定的角色名称为 `Service Administrator`、`Power User`、`User` 或 `Viewer`。

使用 `listfiles` 命令验证文件名及其位置。如果该文件不在收件箱中，则使用 uploadFile 命令上传该文件。

退出代码 6 错误

Service Unavailable 服务由于 HTTP 错误 404 而不可用。

通过从您运行 EPM Automate 的计算机上的浏览器访问服务来验证服务可用性。如果服务由于任何原因而关闭，请等待一会并重试或者与 Oracle 技术支持联系。

Read/Write timeout 如果客户端套接字由于网速慢或者防火墙问题而在任何读/写操作过程中超时（套接字超时为 15 分钟），将显示此错误。

网络吞吐量较高时重新运行失败的命令。如果失败是由于防火墙设置，请与您的网络管理员联系。

退出代码 7 错误

如果 EPM Automate 无法执行命令，它将显示此错误。错误消息（例如 `Invalid command`）指定错误发生原因。
Unable to open password file *FILE_NAME* 加密密码文件无效，例如 *PASSWORD_FILE*.EPW。EPM Automate 在您指定的位置找不到文件或者文件不是所需格式。

验证文件名和路径。如果文件由于格式无效而无法进行解析，请使用 encrypt 命令重新创建该文件。

Unable to parse password file *FILE_NAME* 由于格式无效或者因为加密的密码文件已损坏，EPM Automate 无法解析该文件。

使用 encrypt 命令重新创建该文件。

Unable to connect to *URL*.Root cause *MESSAGE* 如果由于 URL 错误而无法建立连接，将显示此错误。显示根本原因的消息说明了失败根源是使用的 URL 不正确。

- 验证您使用的 URL 是否有效
- 如果您的代理设置要求您在代理服务器中进行验证才能连接 Internet，则指定代理服务器用户名、域和密码（或使用包含代理服务器密码的加密密码文件）来登录。如果需要帮助，请与您的网络管理员联系。

Unable to connect to *URL* Unsupported Protocol 由于指定的 URL 使用不受支持的协议，login 命令失败。随附的错误消息标识了不受支持的协议。

确保您在 login 命令中使用的 URL 采用安全协议 (HTTPS)。

Session is not authenticated.Please execute the "login" command before executing any other command 您试图在使用 `login` 命令建立会话之前运行命令。

运行 login 命令与环境建立安全连接后，再尝试执行其他命令。

Invalid parameter 此消息指出命令中存在用法错误，即命令参数顺序不正确或者缺少一些必需的命令参数值。

检查并更正命令参数及其指定顺序。

COMMAND_NAME* command is not supported by *SERVICE_TYPE EPM Automate 无法针对连接到的环境运行此命令，因为该业务流程不支持此命令。

有关每种业务流程支持的命令列表，请参阅"命令参考"。

File does not exist in location: *PATH* EPM Automate 找不到您要处理的文件（例如使用 `upload` 或 `replay` 命令处理的文件）。

确保文件名和路径正确。

Unable to open file for read: *PATH* EPM Automate 无法从指定文件进行读取。

确保文件采用所需格式。验证运行 EPM Automate 的用户是否对文件具有读取权限。

Unable to open file for write: *PATH* EPM Automate 无法向指定文件进行写入。

确保文件未被其他进程锁定。验证运行 EPM Automate 的用户是否对文件具有写入权限。

Invalid command EPM Automate 遇到不受支持的命令。

验证 EPM Automate 是否支持该命令；确保命令名称拼写正确。

Invalid date format 该工具遇到无效日期格式。

使用支持的日期格式指定报表生成日期。

FROMDATE *DATE* cannot be greater than TODATE *DATE* EPM Automate 遇到结束日期早于开始日期。

确保指定日期范围内的 `to date` 晚于 `from date`。

Exceeded maximum number of feedbacks (6) for a day 超过了使用 `feedback` 命令可以提交的反馈数量时，将显示此错误。

File with the same name already exists in the download path *PATH*.Please delete the file and rerun the command 尝试将文件下载到的位置已存在与正下载的文件的名称相匹配的文件时，将显示此错误。

删除、重命名或移动现有文件，然后重新运行该命令。

File is empty: *PATH* 如果重放文件不包含任何内容，将显示此错误。

确保重放文件（CSV 文件）列出了用于运行 `replay` 命令的凭据（用户名和密码）以及 HAR 文件的名称。

Unable to encrypt the password as localhost cannot be resolved.Ensure that hostnames are mapped properly to IP addresses 如果由于计算机上的主机文件中地址 `127.0.0.1` 的服务器名称不是 `localhost` 而导致 EPM Automate 无法将 localhost 定义解析为 MAC 地址，将显示此错误。

确保主机文件将 `localhost` 指定为 `127.0.0.1` 的服务器名称

Snapshot Name is invalid 如果未指定要重命名的快照的名称，则显示此错误。

指定环境中可用的快照的名称。

New Snapshot Name is invalid 如果没有为快照指定新名称，则显示此错误。

指定快照的新名称。

`Invalid snapshot name: {0}.Characters \\/*?"<>| are not allowed` 如果快照名称包含特殊字符，例如空格、\（反斜杠）、/（正斜杠）、*（星号）、?（问号）、"（双引号）、<（小于）和 >（大于），将显示此错误。

指定不包含这些特殊字符的新快照名称。

`Unable to rename snapshot : {0}.There could be another process accessing it.Please try after sometime` 如果 EPM Automate 由于另一个进程正在使用快照而无法获取快照的独占锁定，将显示此错误。

请等待正在使用快照的当前操作完成，然后重试。

`Snapshot {0} already exist.Please delete the snapshot and re-run the command` 如果新的快照名称与环境中的现有快照的名称相同，将显示此错误。

使用其他快照名称或使用 `deletefile` 命令删除现有快照。

退出代码 9 错误

`Invalid credentials` 用于 login 命令的用户名或密码不正确时，将显示此错误。

为您尝试连接的环境指定有效凭据。

`Authentication failed while executing command.Please retry` 如果在执行 login 以外的命令过程中基本身份验证失败，将显示此错误。重试命令时（最多三次），HTTP 调用也可能会发生此错误。

退出代码 11 错误

Internal server error.Due to manual reset service, your Oracle EPM Cloud Service environment is currently unavailable.如果在重置环境时运行 EPM Automate 命令，将显示此错误。

`Internal server error MESSAGE` 如果 EPM Automate 遇到与 HTTP 连接无关的未知异常，将显示此错误。包括服务器错误 503 和 500。

`Unable to connect to URL: MESSAGE` 服务器不可用时，将显示此错误。错误消息指出导致命令失败的异常。

如果服务器不可用，请与 Oracle 技术支持联系。如果消息指明 URL 存在问题，验证您使用的 URL 是否有效。

3

命令执行示例方案

EPM Automate 可以用来自动执行 Oracle Fusion Cloud Enterprise Performance Management 中的许多常见管理任务。

- 所有服务的示例方案
- Planning、Consolidation、Tax Reporting 和 Enterprise Profitability and Cost Management 的示例方案
- Account Reconciliation 示例方案
- Profitability and Cost Management 示例方案
- Oracle Enterprise Data Management Cloud 示例方案

关于复制示例脚本

请勿从此文档的 PDF 版本中复制示例脚本。为避免换行符和页脚信息使脚本不可用，Oracle 建议从《使用 *EPM Automate*》的 HTML 版本中复制示例脚本。

所有服务的示例方案

这些方案描述了可用于在 Oracle Fusion Cloud Enterprise Performance Management 环境中执行特定操作的典型命令序列。

另请参阅：

- 将应用程序快照备份到计算机
  此方案说明如何自动将日常服务维护过程中创建的快照备份到本地计算机。
- 向用户发送完成日常维护的通知
  Oracle Fusion Cloud Enterprise Performance Management 环境的日常维护所需时间通常远少于标记的一小时。
- 将快照复制到 Oracle Object Storage 或从中复制快照
- 创建用户并为其分配预定义角色
  使用本节中的脚本可创建用户并为其分配身份域中的预定义角色。
- 统计许可的用户数（分配了角色的用户）
  使用本节中的脚本可生成角色分配报表来统计环境的用户数。
- 创建分配了角色的用户的审核报表
  使用本节中的脚本可针对环境中分配了预定义角色的用户自动创建审核报表，并通过电子邮件将报表发送至收件人（可选）。
- 创建角色分配和撤消审核报表
  使用本节中的 PowerShell 脚本可自动创建用于详述环境中角色分配和角色撤消的审核报表。
- 遮蔽访问日志和活动报表以确保符合隐私法
  使用本节中的脚本可自动遮蔽活动报表或访问日志中的信息以确保符合隐私法，并通过电子邮件将报表发送至收件人（可选）。

- 自动将活动报表下载到本地计算机
  使用本节中的脚本可自动将活动报表从环境中下载到本地计算机。
- 从环境中下载访问日志
  使用本节中的脚本可自动将访问日志从环境中下载到本地计算机。
- 自动克隆环境
  使用本节中的脚本可自动克隆环境。
- 在主环境完成日常维护后，每天从主环境克隆到备用环境
  要使备用环境与主环境保持同步，请在主环境完成日常维护后立即使用这些脚本将 Oracle Fusion Cloud Enterprise Performance Management 主环境克隆到备用环境。
- 从环境中删除不必要的文件
  使用这些脚本可从环境中删除不必要的文件。
- 从环境中查找并下载文件
  使用本节中的示例脚本可将文本字符串用作通配符从 Oracle Fusion Cloud Enterprise Performance Management 环境自动下载一个或多个文件。
- 重新创建一个旧云 EPM 环境用于审核
  使用本节中的脚本可创建自助服务解决方案以维护 Oracle Fusion Cloud Enterprise Performance Management 环境的最新快照库。您需要有一个环境，专门用来升级和维护最新快照库。
- 自动进行数据库访问审核与合规性调节
  使用本节中的 PowerShell 和 Bash Shell 脚本可以利用 EPM Automate 命令收集与手动访问数据库相关的审核及合规性数据。
- 复制用户和预定义角色分配
  本节中的脚本有助于将一个环境的用户和预定义角色分配迁移到另一个环境。
- 创建每季度一次的云 EPM 升级节奏
  使用这些脚本可创建用于跳过更新的自助服务解决方案，以便云 EPM 环境按季度更新，并实施为期两周的测试周期。在这种情况下，在更新测试环境两周后更新生产环境。
- 创建每季度一次的云 EPM 升级节奏，并实施为期六周的测试周期
  使用本节中的脚本可创建用于跳过更新的自助服务解决方案，以便 Oracle Fusion Cloud Enterprise Performance Management 环境按季度更新，并实施为期六周的测试周期。在这种情况下，在更新测试环境六周后更新生产环境。

将应用程序快照备份到计算机

此方案说明如何自动将日常服务维护过程中创建的快照备份到本地计算机。

- 下载在维护期间创建的应用程序快照 (`Artifact Snapshot`)
- 通过附加时间戳重命名下载的快照
- 如果需要，通过删除最早的备份保留 10 个备份

> **Note:**
>
> - 此脚本不能用于备份 Narrative Reporting
> - 如果将该脚本改作他用，请根据需要修改运行时参数（`url`、`user`、`password` 和 `NumberOfBackups`）的值。

有关使用 Windows 任务计划程序计划脚本的信息，请参阅“自动执行脚本”。

Windows 示例脚本

创建包含类似以下脚本的批处理 (.bat) 或 shell (.sh) 文件以自动下载快照。

```
@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 示例脚本

创建一个 shell (.sh) 文件，在其中包含类似如下所示的脚本，以自动下载快照。如果密码中包含特殊字符，请参阅“处理特殊字符”。

```
#!/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 --
```

向用户发送完成日常维护的通知

Oracle Fusion Cloud Enterprise Performance Management 环境的日常维护所需时间通常远少于标记的一小时。

环境的实际日常维护持续时间记录为活动报表的 "Operations Metrics"（操作度量）部分的 "Daily Maintenance Duration in Minutes"（日常维护持续时间 (分钟)）度量值。如果您不想等待整整一个小时才能使用环境，可使用自定义版本的此脚本向用户发送完成日常维护的通知，以便他们可以恢复活动。

Windows 脚本

通过复制以下 PowerShell 脚本来创建 daily_maintenance_completed.ps1。有关更新脚本以供使用的信息，请参阅“重新运行该脚本”。

```
# 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 脚本

通过复制以下脚本来创建 daily_maintenance_completed.sh。有关更新脚本以供使用的信息，请参阅“重新运行该脚本”。

```
#!/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."
```

服务器端 Groovy 脚本

通过复制以下代码来创建 `daily_maintenance_completed` Groovy 脚本。有关更新脚本以供使用的信息，请参阅“重新运行该脚本”。

```
// 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.")
```

重新运行该脚本

Windows 和 Linux/UNIX

1. 通过复制上一节的脚本来创建 `daily_maintenance_completed.ps1` 或 `daily_maintenance_completed.sh`。
2. 更新脚本：
   - **Windows**：将 `emailaddresses` 值更新为在完成日常维护时应通知的电子邮件地址列表（以逗号分隔）。
   - **Linux/UNIX**：更新以下变量：
     - 将 `epmautomatescript` 更新为 EPM Automate 可执行文件的位置。示例：`epmautomatescript="/home/utils/EPMAutomate/bin/epmautomate.sh"`
     - 将 `javahome` 更新为 EPM Automate 使用的 JDK 的安装目录。例如：`"/home/user1/jdk1.8.0_191"`
     - 将 `emailaddresses` 更新为在完成日常维护时应通知的电子邮件地址列表（以逗号分隔）。例如：`jdoe@example.com,jane_doe@example.com`
3. 在命令窗口或控制台中，导航到存储 `daily_maintenance_completed` 脚本的文件夹。
4. 运行以下命令：
   - **Windows**：`./daily_maintenance_completed.ps1 *USERNAME PASSWORD URL*`
   - **Linux/UNIX**：`./daily_maintenance_completed.sh *USERNAME PASSWORD URL*`，其中：
     - `USERNAME` 是服务管理员的用户名
     - `PASSWORD` 是服务管理员的密码
     - `URL` 是云 EPM 环境的 URL

服务器端 Groovy：

1. 通过复制上一节的脚本来创建 `daily_maintenance_completed.groovy` Groovy 脚本。
2. 更新以下值。
   - 将 `username` 更新为服务管理员的用户名。

- 将 password 更新为服务管理员的密码
- 将 url 更新为需要对其发出完成日常维护通知的云 EPM 环境的 URL。例如：示例：https://testExample-idDomain.pbcs.us1.oraclecloud.com
- 将 emailaddresses 更新为在完成日常维护时应通知的电子邮件地址列表（以逗号分隔）。

3. 在云 EPM 业务流程中使用 Groovy 屏幕，或使用 runBusinessRule 自动执行脚本。有关详细信息，请参阅以下信息源：
   - 在不安装 EPM Automate 的情况下运行命令
   - 《管理 *Planning*》中的“使用 Groovy 规则”

将快照复制到 Oracle Object Storage 或从中复制快照

本主题包含用于完成以下任务的示例脚本：

- 将 Artifact Snapshot（维护快照）从 Oracle Fusion Cloud Enterprise Performance Management 复制到 Oracle Object Storage 存储桶，并通过附加快照复制日期来重命名快照。
- 将备份快照从 Oracle Object Storage 存储桶复制到云 EPM。

此部分中的脚本假定您已经在 Oracle Object Storage 中创建了用于存放快照的存储桶。在运行这些脚本之前，请通过更新以下占位符来自定义这些脚本，使其适合您使用：

Table 3-1 参数及其值

占位符	预期值
JAVA_HOME	EPM Automate 使用的 JDK 的安装目录。 **示例：**./home/JDK/bin
epmautomateExe	EPM Automate 的安装目录。 **示例：**./home/utils/EPMAutomate/bin
cloudServiceUser	云 EPM 服务管理员的用户 ID。 **示例：**John.doe@example.com
cloudServicePassword	服务管理员的密码，或者密码文件的位置。如果密码中包含特殊字符，请参阅“处理特殊字符”。 **示例：**ex_PWD_213
cloudServiceUrl	要从中复制 Artifact Snapshot 的云 EPM 环境的 URL。 **示例：**https//test-cloud-id_Dom.pbcs.us1.oraclecloud.com
objectStorageUser	Oracle Object Storage 中用户的用户 ID。 要将快照复制到对象存储，此用户必须对向其中复制快照的存储桶具有写入访问权限。要从对象存储复制快照，此用户必须对从中复制快照的存储桶具有读取访问权限。 **示例：**jDoe
objectStoragePassword	objectStorageUser 的密码。 **示例：**example_PWD

Table 3-1 (Cont.) 参数及其值

占位符	预期值
objectStorageBucketUrl	要将快照复制到其中的 Oracle Object Storage 存储桶的 URL。请参阅以下信息来源，了解 URL 格式： • copyToObjectStorage • copyFromObjectStorage **示例：** https//swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/axaxnpcrorw5/bucket-20210301-1359
snapshot	要从 Oracle Object Storage 存储桶复制的快照的名称。 **示例：** Artifact Snapshot20210429.zip

将快照从云 EPM 复制到 Oracle Object Storage 的示例 EPM Automate 脚本

自定义并运行此脚本以对其重命名，然后将 Artifact Snapshot 从云 EPM 复制到 Oracle Object Storage 存储桶。

```
#!/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
```

将快照从 Oracle Object Storage 复制到云 EPM 的示例 EPM Automate 脚本

自定义并运行此脚本，以将特定日期的 Artifact Snapshot 从 Oracle Object Storage 存储桶复制到云 EPM。

```
#!/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
```

创建用户并为其分配预定义角色

使用本节中的脚本可创建用户并为其分配身份域中的预定义角色。

这些脚本使用 EPM Automate 命令完成以下活动：

- 以具有身份域管理员角色的服务管理员身份登录环境。
- 从环境导出组和成员身份信息以重新生成您指定的快照，例如 `example_snapshot.zip`。假设您先前使用迁移导出组和成员身份对象创建了此快照。
- 将快照 (`example_snapshot.zip`) 下载到本地目录。
- 从环境中删除快照 (`example_snapshot.zip`)。
- 从环境中注销。
- 执行为其添加了自定义代码的操作。此类操作可以包括：
  - 提取 `example_snapshot.zip` 的内容
  - 以“名字,姓氏,电子邮件,用户登录名”格式将新用户信息附加到 `HSS-Shared Services\resource\External Directory\Users.csv`。有关详细信息，请参阅《*Getting Started with Oracle Cloud*》中的 "Importing a Batch of User Accounts"。
  - 将新用户的角色分配信息（以“名字,姓氏,电子邮件,用户登录名”格式）附加到 `HSS-Shared Services\resource\External Directory\Roles\` 中的适当角色文件。例如，服务管理员角色分配应附加到 `<service_name> Service Administrator.csv`，而查看者角色分配应附加到 `HSS-Shared Services\resource\External Directory\Roles\<service_name> Viewer.csv`。有关详细信息，请参阅《*Getting Started with Oracle Cloud*》中的 "Assigning One Role to Many Users"。
  - 通过压缩 `HSS-Shared Services` 目录及其内容，使用更新的文件重新创建快照。
- 以同时具有身份域管理员角色的服务管理员身份登录到环境。
- 将修改后的 `example_snapshot.zip` 上传到环境。
- 将 `example_snapshot.zip` 导入环境中。
- 从环境中删除上传的 `example_snapshot.zip`。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建一个名为 `createUsersAndAssignRoles.ps1` 的文件。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建一个名为 `createUsersAndAssignRoles.sh` 的文件。将其存储在本地目录中。

```
#!/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
```

示例 input.properties 文件

要运行 createUsersAndAssignRoles 脚本，请创建 input.properties 文件并使用环境信息进行相应的更新。将文件保存在存储 createUsersAndAssignRoles.ps1 或 createUsersAndAssignRoles.sh 的目录中。

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
```

表 3-2 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
snapshotName	要生成的快照的名称。假设您先前使用迁移导出组和成员身份对象创建了此快照。
userspassword	新用户的初始密码。
resetPassword	新用户初次登录时是否必须重置密码。将此值设置为 true 将强制新用户在首次登录时更改其密码。

重新运行该脚本

1. 通过复制上一节的脚本来创建 createUsersAndAssignRoles.ps1 或 createUsersAndAssignRoles.sh。
2. 添加自定义代码以执行以下操作：
   - 提取快照的内容

- 将新用户信息附加到 `HSS-Shared Services\resource\External Directory\Users.csv`。
- （可选）将新用户的角色分配信息（以“名字,姓氏,电子邮件,用户登录名”格式）附加到 `HSS-Shared Services\resource\External Directory\Roles\` 中的适当角色文件。
- 使用更新的文件重新创建快照。

3. 创建 `input.properties` 文件，并将其保存在 `createUsersAndAssignRoles` 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“示例 input.properties 文件”。
请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
4. 启动脚本。
   - **Windows PowerShell**：运行 `createUsersAndAssignRoles.ps1`。
   - **Linux/UNIX**：运行 `./createUsersAndAssignRoles.sh`。

统计许可的用户数（分配了角色的用户）

使用本节中的脚本可生成角色分配报表来统计环境的用户数。

通过复制以下脚本来创建 `provisionedUsersCount.bat`。

> **注：**
>
> - 用于运行 `provisionedUsersCount.bat` 的输入参数为 `username`、`password/password_file`、`service_url` 和 `report_file_name`。例如，在命令提示符下，输入类似下面的命令：
> `provisionedUsersCount.bat jdoe password.epw https://example.oraclecloud.com myRole_assign.CSV`
> - 如果密码中包含特殊字符，请参阅“处理特殊字符”。

```
@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 "%4"
if %errorlevel% NEQ 0 exit /b 1
   call epmautomate.bat downloadFile "%4"
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%.
)
```

创建分配了角色的用户的审核报表

使用本节中的脚本可针对环境中分配了预定义角色的用户自动创建审核报表，并通过电子邮件将报表发送至收件人（可选）。

此审核报表显示分配了预定义角色的用户或上次生成报表后发生变化的组。要创建每日审核报表，请每天运行此脚本。

通过复制以下脚本来创建 `provisioningAuditReport.bat`。此包装器批处理脚本调用 PowerShell 脚本 `provisioningAuditReport.ps1`，此方案后面提供了其源代码。

> **注：**
>
> - 用于运行 `provisioningAuditReport.bat` 的输入参数为：`username`、`password` 或 `password_file`、`service_url` 以及 `report_email_to_address`（可选，仅当您要将报表发送到电子邮件地址时才需要）。
> - 如果密码中包含特殊字符，请参阅“处理特殊字符”。

```
@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 调用 provisioningAuditReport.ps1，您通过复制以下脚本创建后者。

provisioningAuditReport.ps1 创建审核报表。将其放在 provisioningAuditReport.bat 所在的同一目录中。

```
$username=$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"
```

创建角色分配和撤消审核报表

使用本节中的 PowerShell 脚本可自动创建用于详述环境中角色分配和角色撤消的审核报表。

通过复制以下脚本来创建 `AuditReportRoleAssignment.bat`。此包装器批处理脚本调用 PowerShell 脚本 `AuditReportRoleAssignment.ps1`，此方案后面提供了其源代码。

> **注：**
>
> - 用于运行 `AuditReportRoleAssignment.bat` 的输入参数为 `username`、`password` 或 `password_file` 和 `service_url`。
> - 如果密码中包含特殊字符，请参阅“处理特殊字符”。

脚本：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 %*
```

脚本：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"
```

遮蔽访问日志和活动报表以确保符合隐私法

使用本节中的脚本可自动遮蔽活动报表或访问日志中的信息以确保符合隐私法，并通过电子邮件将报表发送至收件人（可选）。

由于某些国家/地区的隐私法较严格，活动报表和访问日志中可用的信息可能必须对服务管理员隐藏以保护用户的隐私。

可以使用 `anonymizeData.bat` 遮蔽活动报表或访问日志中的信息以确保符合隐私法，并通过电子邮件发送报表（可选）。要遮蔽信息，请使用 Windows 计划程序调度此脚本或其变体，让它每日在每个环境的日常维护流程完成后很快运行。

使用以下信息源：

- 《管理员入门指南》中的“使用活动报表和访问日志来监视使用情况”
- 自动执行脚本

通过复制以下过程中提供的 Windows 脚本并使用 Windows 计划程序调度该脚本来手动创建 `anonymizeData.bat`。如果不想使用 Windows 进行调度，也可以创建和运行适合平台的类似脚本。

`anonymizeData.bat` 是一种包装器脚本，可以执行 `anonymizeData.ps1` 脚本，您可以按照以下过程中的说明创建并更新该脚本。

如果密码中包含特殊字符，请参阅“处理特殊字符”

1. 创建一个名为 anonymizeData.bat 的批处理 (BAT) 文件，在其中包含以下脚本，并将它保存到一个方便的位置，例如 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. 创建一个名为 anonymizeData.ps1 的 PowerShell 脚本 (PS1) 文件，在其中包含以下脚本，并将它保存到一个方便的位置，例如 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. 使用 Windows 计划程序调度 `anonymizeData.bat`。有关详细步骤，请参阅“自动执行脚本”。

   您需要提供以下参数值以执行 `anonymizeData.bat`

   - 服务管理员的用户名
   - 服务管理员的密码或提供加密密码文件的位置
   - 必须遮蔽访问日志和活动报表的服务环境的 URL
   - **可选：** 要将报表发送到的电子邮件地址。仅当指定此值时，才会通过电子邮件发送报表。

自动将活动报表下载到本地计算机

使用本节中的脚本可自动将活动报表从环境中下载到本地计算机。

可以使用 `syncAprReports.bat` 下载活动报表。您可以使用 Windows 计划程序来调度批处理文件以自动执行活动报表的下载。有关活动报表的详细信息，请参阅《管理员入门指南》中的“使用活动报表和访问日志来监视使用情况”。

您可以手动创建 `syncAprReports.bat`，方法是复制以下过程中提供的脚本，然后更新连接参数。此脚本将检查环境，且只下载比本地计算机下载目录中的报表更新的报表。

> **注：**
>
> - 此脚本只能从 Windows 计算机上运行。
> - 此脚本不会下载用户提交反馈时生成的反馈活动报表。
> - 如果密码中包含特殊字符，请参阅“处理特殊字符”

1. 创建一个名为 `syncAprReports.bat` 的批处理 (`.BAT`) 文件（其中包含以下脚本），并将其保存到一个方便的位置，例如 `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. 修改 `syncAprReports.bat` 以设置下表中参数的值。这些值用于访问环境以下载活动报表。

表 3-3 要在 `syncAprReports.bat` 中包含的参数值

参数	预期值
`set apr_dir=%1`	指定要将活动报表下载到的现有目录。 **示例：** `set apr_dir="C:\Oracle\apr"`
`set username=%2`	用于登录到环境下载活动报表的 Oracle Fusion Cloud Enterprise Performance Management 用户名。 **示例：** `set username="ServiceAdmin"`

表 3-3 （续） 要在 syncAprReports.bat 中包含的参数值

参数	预期值

`set password=%3`	用于存储由 `username` 变量指定的用户的加密密码的文件的名称和位置。您也可以指定用户的明文密码（不建议）。有关创建加密密码文件的信息，请参阅 encrypt 命令。
	示例：
	`set password="C:\mySecuredir\password.epw"`
	`set password="Ex@mple1"`

`set url=%4`	环境的 URL。 **示例：** `set url="https://test-example.stg-pbcs.us1.oraclecloud.com"`

3. 使用 Windows 计划程序，调度 `syncAprReports.bat`。有关详细步骤，请参阅“自动执行脚本”。

从环境中下载访问日志

使用本节中的脚本可自动将访问日志从环境中下载到本地计算机。

您可以使用 Windows 计划程序来调度 `syncAccessLog.bat` 以自动执行日志文件的下载。有关使用应用程序管理下载访问日志的过程，请参阅《管理员入门指南》中的“查看和下载活动报表和访问日志”。

以下脚本将检查环境，且只下载比本地计算机下载目录中的日志文件更新的日志文件。这是 Windows 脚本；您可以为 Linux/UNIX 环境创建相似的 shell 脚本。

1. 创建一个名为 `syncAccessLog.bat` 的批处理 (.BAT) 文件（其中包含以下脚本），并将其保存到一个方便的位置，例如 `C:\automate_scripts`。

> **注：**
>
> 如果密码中包含特殊字符，请参阅“处理特殊字符”。

```
@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. 修改 `syncAccessLog.bat` 以设置下表中参数的值。这些值用于访问环境以下载访问日志。

表 3-4　要在 syncAccessLog.bat 中包含的变量值

变量	预期值
`set apr_dir=%1`	指定要将访问日志下载到的目录。 **示例：** `set apr_dir="C:\Oracle\apr"`
`set username=%2`	用于登录到环境下载访问日志的 Oracle Fusion Cloud Enterprise Performance Management 用户名。 **示例：** `set username="ServiceAdmin"`

`set password=%3`	用于存储由 `username` 变量指定的用户的加密密码的文件的名称和位置。您也可以指定用户的明文密码（不建议）。有关创建加密密码文件的信息，请参阅 encrypt 命令。
	示例：
	`set password="C:\mySecuredir\password.epw"`
	`set password="P@ssword1"`

`set url=%4`	环境的 URL。 **示例：** `set url="https://test-cloudpln.pbcs.us1.oraclecloud.com"`

3. 使用 Windows 计划程序，调度 `syncAccessLog.bat`。有关详细步骤，请参阅“自动执行脚本”。

自动克隆环境

使用本节中的脚本可自动克隆环境。

创建包含类似以下脚本的批处理 (`.bat`) 或 shell (`.sh`) 文件以克隆环境。后面的示例脚本可以处理如下活动：

- 登录到源环境。
- 使用 `Artifact Snapshot`（在源环境的上次日常维护期间创建的快照）或源环境中可用的其他快照，将目标环境转换为源环境的克隆。
- （可选）创建用户及其预定义角色和应用程序角色分配，以与源环境中的这些用户及其分配匹配。
- （可选）更改日常维护开始时间以与源环境的该时间匹配。
- 注销。

有关克隆过程的详细信息，请参阅《管理迁移》中的“克隆云 EPM 环境 ”。

有关使用 Windows 任务计划程序计划脚本的信息，请参阅“自动执行脚本”。

Windows

1. 创建一个名为 `cloneEnvironment.bat` 的批处理 (.BAT) 文件（其中包含以下脚本），并将其保存到一个方便的位置，例如 `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. 修改 `cloneEnvironment.bat` 以设置以下参数的值：

表 3-5　要在 cloneEnvironment.bat 中设置的参数

参数	说明
`usersandpredefinedroles`	将此参数的值设置为 `true` 可克隆用户及其预定义角色和应用程序角色分配。 要克隆用户和角色分配，运行该脚本的用户必须在目标环境中具有服务管理员和身份域管理员角色。
`snapshotname`	源环境中应该用于克隆的快照的名称。
`dailymaintenancestarttime`	将此参数的值设置为 `true` 可将克隆环境的日常维护开始时间更改为源环境的该时间。将此值设置为 `false` 可保留克隆环境的当前日常维护开始时间。

3. 创建一个名为 `cloneEnvironment.ps1` 的 PowerShell 脚本（其中包含以下脚本），并将其保存到您保存 `cloneEnvironment.bat` 的目录，例如 `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. 使用以下命令运行 `cloneEnvironment.bat`：

```
cloneEnvironment.bat "SOURCE USERNAME" "SOURCE PASSWORD FILE" "SOURCE URL"
"TARGET USERNAME" "TARGET PASSWORD FILE" "TARGET URL"
```

例如：

```
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. 创建一个名为 `cloneEnvironment.sh` 的 shell 脚本（其中包含以下脚本），并将其保存到一个方便的位置。

```
#!/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. 修改 `cloneEnvironment.sh` 以设置以下参数的值：

表 3-6　要在 cloneEnvironment.sh 中设置的参数

参数	说明
`epmautomatescript`	EPM Automate 可执行文件 (`epmautomate.sh`) 的绝对路径。
`javahome`	JAVA_HOME 位置。
`usersandpredefinedroles`	将此参数的值设置为 `true` 可克隆用户及其预定义角色和应用程序角色分配。 要克隆用户和角色分配，运行该脚本的用户必须在目标环境中具有服务管理员和身份域管理员角色。
`snapshotname`	源环境中应该用于克隆的快照的名称。
`dailymaintenancestarttime`	将此参数的值设置为 `true` 可将克隆环境的日常维护开始时间更改为源环境的该时间。将此值设置为 `false` 可保留克隆环境的当前日常维护开始时间。

3. 运行 `cloneEnvironment.sh`。

```
./cloneEnvironment.sh "SOURCE USERNAME" "SOURCE PASSWORD FILE" "SOURCE
URL" "TARGET USERNAME" "TARGET PASSWORD FILE" "TARGET URL"
```

例如：

```
./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.
```

在主环境完成日常维护后，每天从主环境克隆到备用环境

要使备用环境与主环境保持同步，请在主环境完成日常维护后立即使用这些脚本将 Oracle Fusion Cloud Enterprise Performance Management 主环境克隆到备用环境。

这些自定义脚本标识当天安排的日常维护是否完成，然后克隆环境。

Windows 脚本

通过复制以下 PowerShell 脚本来创建 `dailyclone.ps1`。

```
# 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 脚本

通过复制以下脚本来创建 `dailyclone.sh`。

```
#!/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."
```

Groovy 脚本

通过复制以下代码来创建 `dailyclone` groovy 脚本。

```
// Clone Environment script

// Update the following parameters
String source_username="USERNAME"
String source_password="PASSWORD!"
String source_url="URL OF THE SOURCE ENVIRONMENT"
String target_username="USERNAME"
String target_password="PASSWORD"
String target_url="URL OF THE TARGET ENVIRONMENT"
String snapshot_name="Artifact Snapshot"

// -------------------------------

// Do not update the following parameters
String users_and_predefined_roles="false"
String data_management="true"
String app_audit="true"
String job_console="true"
String stored_snapshots_and_files="false"
String daily_maintenance_start_time="true"
// -------------------------------

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)
}

boolean artifactSnapshotExists(EpmAutomate automate,String snapshot_name) {
    LogMessage("Operation: listfiles")
    EpmAutomateStatus listfilesstatus = automate.execute('listfiles')
    String filelist=(listfilesstatus.getItemsList())
    LogOperationStatus(listfilesstatus)

    if (filelist.contains(snapshot_name)) {
        return true
    } else {
        return false
    }
}

def deleteArtifactSnapshotIfExists(EpmAutomate automate,String snapshot_name){
    if (artifactSnapshotExists(automate,snapshot_name)) {
        LogMessage("Operation: deletefile " + snapshot_name)
        EpmAutomateStatus deletefilestatus =
automate.execute('deletefile',snapshot_name)
```

```
        LogOperationStatus(deletefilestatus)
    }
}

def waitForDailyMaintenanceToComplete(EpmAutomate automate,String
snapshot_name) {
    LogMessage("Checking to see if daily maintenance processing has
completed ...")
    while (true) {
        if (artifactSnapshotExists(automate,snapshot_name)) {
            LogMessage("Daily maintenance processing has completed ...")
            break
        } else {
            LogMessage("Sleeping for 30 seconds before the next check to see
if daily maintenance processing has completed ...")
            sleep(30000)
        }
    }
}

LogMessage("Clone environment script processing beginning ...")

EpmAutomate automate = getEpmAutomate()

LogMessage("Operation: login " + source_username + " " + source_password + "
" + source_url)
EpmAutomateStatus status =
automate.execute('login',source_username,source_password,source_url)
LogOperationStatus(status)

String amwtime = getDailyMaintenanceStartTime(automate)
goToSleep (amwtime)
deleteArtifactSnapshotIfExists(automate,snapshot_name)
waitForDailyMaintenanceToComplete(automate,snapshot_name)

LogMessage("Operation: encrypt " + target_password + " oracleKey
target_password.epw")
status =
automate.execute('encrypt',target_password,'oracleKey','target_password.epw')
LogOperationStatus(status)

LogMessage("Operation: cloneEnvironment " + target_username + "
target_password.epw " + target_url + " SnapshotName=" + snapshot_name + "
UsersAndPreDefinedRoles=" + users_and_predefined_roles + " DataManagement=" +
data_management + " appAudit=" + app_audit + " jobConsole=" + job_console + "
storedSnapshotsAndFiles=" + stored_snapshots_and_files + "
dailyMaintenanceStartTime=" + daily_maintenance_start_time)
EpmAutomateStatus cloneenvironmentstatus =
automate.execute('cloneEnvironment',target_username,"target_password.epw",targ
et_url,"SnapshotName=" + snapshot_name,"UsersAndPreDefinedRoles=" +
users_and_predefined_roles,"DataManagement=" + data_management,"appAudit=" +
app_audit,"jobConsole=" + job_console,"storedSnapshotsAndFiles=" +
stored_snapshots_and_files,"dailyMaintenanceStartTime=" +
daily_maintenance_start_time)
LogOperationStatus(cloneenvironmentstatus)
```

```
LogMessage("Operation: logout ")
status = automate.execute('logout')
LogOperationStatus(status)

LogMessage ("Clone environment script processing has completed.")
```

重新运行该脚本

1. 通过复制前面几节之一中的脚本来创建 `dailyclone.ps1`、`dailyclone.sh` 或 `dailyclone` groovy 脚本。
2. 更新脚本：
   `dailyclone.sh` 的更新：
   - 将 `epmautomatescript` 更新为 EPM Automate 可执行文件的位置。示例：`epmautomatescript="/home/utils/EPMAutomate/bin/epmautomate.sh"`
   - 将 `javahome` 更新为 EPM Automate 使用的 JDK 的安装目录。例如：`"/home/user1/jdk1.8.0_191"`

   `dailyclone.ps1` 和 `dailyclone.sh` 的更新：
   - `users_and_predefined_roles`：将此值设置为 `true` 以克隆用户及其预定义角色分配（始终克隆访问控制组）。
   - `data_management`：将此值设置为 `false` 以不克隆数据集成记录。请注意，仅当源环境和目标环境的每月更新相同或目标环境比源环境多更新一次时，才可以克隆数据集成记录。例如，22.01 数据管理记录可以克隆到其他 22.01 环境，或者仅克隆到 22.02 环境。
     对于 Narrative Reporting 和 Oracle Fusion Cloud Enterprise Data Management 环境，忽略此项。
   - `app_audit`：如果不希望克隆 Planning、自由形式和 Enterprise Profitability and Cost Management 应用程序的应用程序审核数据，请将此值设置为 `false`。
     始终克隆 Financial Consolidation and Close 和 Tax Reporting 审核信息。
   - `job_console`：如果不希望克隆作业控制台数据，请将此值设置为 `false`。
   - `stored_snapshots_and_files`：如果要克隆源环境的收件箱和发件箱中顶层文件夹的内容（从不克隆子文件夹），请将此值设置为 `true`。
   - `daily_maintenance_start_time`：如果不希望将克隆的目标环境的维护开始时间重置为源环境的维护开始时间，请将此值设置为 `false`。

   `dailyclone` groovy 脚本的更新：
   - `source_username` 是服务管理员的用户名。身份域管理员角色是克隆用户和预定义角色所必需的。
   - `source_password` 是 *`SOURCE_USERNAME`* 标识的用户的密码。
   - `source_url` 是要克隆的环境的 URL。
   - `target_username` 是服务管理员的用户名。身份域管理员角色是克隆用户和预定义角色所必需的。
   - `target_password` 是 *`TARGET_USERNAME`* 标识的用户的密码。
   - `target_URL` 是目标环境的 URL。
3. 运行 `dailyclone.ps1`、`dailyclone.sh` 或 `dailyclone` groovy 脚本：
   Windows
   - 在命令窗口中，导航到存储 `dailyclone.ps1` 的文件夹。

- 执行命令：`./dailyclone.ps1 SOURCE_USERNAME SOURCE_PASSWORD SOURCE_URL TARGET_USERNAME TARGET_PASSWORD TARGET_URL`。有关这些值的说明，请参阅下表。

Table 3-7 参数说明

参数	说明
SOURCE_USERNAME	服务管理员的用户名。此外，此用户必须具有身份域管理员角色才能克隆用户和预定义角色。
SOURCE_PASSWORD	*SOURCE_USERNAME* 标识的用户的密码。
SOURCE_URL	要克隆的环境的 URL。
TARGET_USERNAME	目标环境中的服务管理员的用户名。此外，此用户必须具有身份域管理员角色才能克隆用户和预定义角色。
TARGET_PASSWORD	*TARGET_USERNAME* 标识的用户的密码。
TARGET_URL	目标环境的 URL。

Linux/UNIX

- 在 UNIX 或 Linux shell 中，导航到存储 `dailyclone.sh` 的目录。
- 执行命令：`./dailyclone.sh SOURCE_USERNAME SOURCE_PASSWORD SOURCE_URL TARGET_USERNAME TARGET_PASSWORD TARGET_URL`。有关这些值的说明，请参阅上表。

Groovy
请参阅“在不安装 EPM Automate 的情况下运行命令”。

从环境中删除不必要的文件

使用这些脚本可从环境中删除不必要的文件。

这些脚本执行以下步骤：

- 登录环境。
- 列出环境中的文件和快照。
- 删除 `input.properties` 中指定的文件。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建一个名为 `removeUnnecessaryFiles.ps1` 的文件。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建一个名为 `removeUnnecessaryFiles.sh` 的文件。将其存储在本地目录中。

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} listfiles
${epmautomatescript} deletefile "${file1}"
${epmautomatescript} deletefile "${file2}"
${epmautomatescript} logout
```

创建 input.properties 文件

要运行 `removeUnnecessaryFiles` 脚本，请创建 `input.properties` 文件并使用环境信息进行相应的更新。将文件保存在存储 `removeUnnecessaryFiles.ps1` 或 `removeUnnecessaryFiles.sh` 的目录中。

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
```

表 3-8 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
file1 和 file2	要从环境中删除的文件或快照的名称。如果文件不在发件箱中，请指定文件的路径和名称。

重新运行该脚本

1. 通过复制上一节的脚本来创建 `removeUnnecessaryFiles.ps1` 或 `removeUnnecessaryFiles.sh`。
2. 创建 `input.properties` 文件，并将其保存在 `removeUnnecessaryFiles` 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 `removeUnnecessaryFiles.ps1`。
   - **Linux/UNIX**：运行 `./removeUnnecessaryFiles.sh`。

从环境中查找并下载文件

使用本节中的示例脚本可将文本字符串用作通配符从 Oracle Fusion Cloud Enterprise Performance Management 环境自动下载一个或多个文件。

使用以下脚本，您可以将您指定为 `FILENAME` 参数值的字符串与使用 `listfiles` 命令显示的文件名进行匹配，然后自动下载与该字符串匹配的文件。

确保将适当的搜索字符串分配给 `FILENAME` 参数。例如，`FILENAME="Scheduler Output/epm"` 将字符串 `Scheduler Output/epm` 与 `listfiles` 命令在环境中输出的文件名进行匹配，以确定要下载的文件。

用于运行此脚本的输入参数为：`username`、`password` 或 `password_file` 和 `service_url`。

> **注：**
>
> 如果密码中包含特殊字符，请参阅“处理特殊字符”。

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
```

重新创建一个旧云 EPM 环境用于审核

使用本节中的脚本可创建自助服务解决方案以维护 Oracle Fusion Cloud Enterprise Performance Management 环境的最新快照库。您需要有一个环境，专门用来升级和维护最新快照库。

云 EPM 仅支持在一个每月周期内快照具有兼容性；您可将维护快照从测试环境迁移到生产环境，也可以执行相反的迁移。但是，某些客户的审核要求可能需要在最新环境中还原多年的快照并在短期内访问应用程序。

应将此脚本计划为每月运行一次，以转换可用快照，并使其与最新的云 EPM 修补程序级别兼容。Oracle 建议您在当月第三个星期五之后运行该脚本，以确保生产环境中的所有问题均已解决。

> **注：**
>
> 您不能使用此脚本更新 Narrative Reporting、Account Reconciliation 和 Oracle Fusion Cloud Enterprise Data Management 快照。

脚本的工作原理

对于客户存储的每个快照，升级脚本都会使用 EPM Automate 完成以下任务：

1. 使用 `input.properties` 文件中的信息登录到环境中
2. 使用 `recreate` 命令刷新环境
3. 将快照导入到环境
4. 对环境执行日常维护，这样可将快照转换为与当前云 EPM 修补程序级别兼容的格式。
5. 将 `Artifact Snapshot`（维护快照）下载到一个文件夹中。如果通过从 `snapshots/18.05` 上传快照重新创建了 18.05 环境，`Artifact Snapshot` 将下载到 `snapshots/18.06` 中。
6. 如果指定，则通过电子邮件将重新创建旧环境的结果发送到电子邮件地址。

重新运行该脚本

1. 创建 input.properties 文件，并使用环境信息进行相应的更新。将该文件保存到一个本地目录。在本次讨论中，此目录名为 parentsnapshotdirectory。此文件的内容因操作系统的不同而异。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
2. 创建 upgradeSnapshots.ps1 (Windows PowerShell) 或 upgradeSnapshots.sh (Linux/UNIX) 脚本，并将其保存到 input.properties 所在的 parentsnapshotdirectory 中。
3. 在 parentsnapshotdirectory 中创建一个子目录，例如 snapshots。
4. 在上一步中创建的目录 (snapshots) 中创建一个子目录，用它来存储您希望进行转换以与当前云 EPM 修补程序级别相兼容的每月快照。使用 YY.MM 格式命名该目录；例如，目录 18.05 用于存储 2018 年 5 月的快照。
5. 将快照复制到相应的子目录。例如，将 2018 年 5 月的快照复制到 snapshots/18.05。
6. 启动脚本。
   - Linux/UNIX：运行 ./upgradeSnapshots.sh。
   - Windows PowerShell：运行 upgradeSnapshots.ps1。

Windows

通过复制本节中的脚本，创建 input.properties 和 upgradeSnapshots.ps1 脚本。

创建 input.properties

```
username=exampleAdmin
userpassword=examplePassword
serviceurl=exapleURL
proxyserverusername=proxyServerUserName
proxyserverpassword=proxyPassword
proxyserverdomain=proxyDoamin
parentsnapshotdirectory=C:/some_directory/snapshots
emailtoaddress=exampleAdmin@oracle.com
```

更新 input.properties

> **注：**
>
> 如果您未在 Windows 网络环境中启用通过代理服务器进行身份验证，请从 input.properties 文件中删除属性 proxyserverusername、proxyserverpassword 和 proxyserverdomain。

表 3-9 input.properties 参数

参数	说明
username	服务管理员的用户名。
userpassword	服务管理员的密码。

表 3-9 （续） input.properties 参数

参数	说明
serviceurl	用于执行此活动的环境的 URL。
proxyserverusername	使用具有 Internet 访问控制权的代理服务器对安全会话进行身份验证的用户名。
proxyserverpassword	用于在代理服务器中对用户进行身份验证的密码。
proxyserverdomain	为代理服务器定义的域的名称。
parentsnapshotdirectory	用于存储待处理快照的目录的父目录的绝对路径。请使用正斜杠 (/) 作为目录分隔符。
emailtoaddress	（可选）要将重新创建旧环境的结果发送到的电子邮件地址。仅当指定此值时，才会通过电子邮件发送结果。 **示例：** john.doe@example.com

> **注：**
>
> 如果密码中包含特殊字符，请参阅“处理特殊字符”。

创建 upgradeSnapshots.ps1

使用此示例脚本创建 upgradeSnapshots.ps1

```
# Script for recreating an old Cloud EPM 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 $emailtoaddress "Recreating An Old Cloud EPM
Environment results" Body="The results of recreating an old Cloud EPM
Environment are attached." Attachments=$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

通过复制以下脚本来创建 `upgradeSnapshots.sh` 和 `input.properties`。

为 Linux/UNIX 创建 `input.properties`

> **注：**
>
> 如果您的网络未配置为使用代理服务器来访问 Internet，请从文件 `input.properties` 中删除 `proxyserverusername`、`proxyserverpassword` 和 `proxyserverdomain` 属性。

```
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
```

更新 `input.properties`

表 3-10　input.properties 参数

参数	说明
`username`	服务管理员的用户名。
`userpassword`	服务管理员的密码。
`serviceurl`	用于执行此活动的环境的 URL。
`proxyserverusername`	使用具有 Internet 访问控制权的代理服务器对安全会话进行身份验证的用户名。
`proxyserverpassword`	用于在代理服务器中对用户进行身份验证的密码。
`proxyserverdomain`	为代理服务器定义的域的名称。
`jdkdir`	`JAVA_HOME` 位置。
`epmautomatescript`	EPM Automate 可执行文件 (`epmautomate.sh`) 的绝对路径。
`parentsnapshotdirectory`	用于存储待处理快照的目录的父目录的绝对路径。
`emailtoaddress`	（可选）要将重新创建旧环境的结果发送到的电子邮件地址。

> **注:**
>
> 如果密码中包含特殊字符，请参阅“处理特殊字符”。

创建 upgradeSnapshots.sh

使用此示例脚本创建 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 $emailtoaddress "Recreating An Old
Cloud EPM Environment results" Body="The results of recreating an old Cloud
EPM Environment are attached" Attachments=$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
```

自动进行数据库访问审核与合规性调节

使用本节中的 PowerShell 和 Bash Shell 脚本可以利用 EPM Automate 命令收集与手动访问数据库相关的审核及合规性数据。

可使用这些脚本完成以下任务：

- 下载当天的活动报表
- 解析报表以确定是否针对环境报告了手动数据库访问信息
- 相对于执行脚本的目录创建 `./reports/dataAccessAuditReport.txt`。该报表列出了访问数据库的时间以及已执行的 SQL 命令。此报表是一种累计型文件，会将最新信息显示在顶部。提供的信息包括：
  - 生成报表的日期和时间
  - 数据库访问详细信息（如果有）。在单独的部分列出了没有服务请求的数据库访问和有服务请求的数据库访问。
    如果活动报表中没有报告手动数据库访问信息，则报表会指出 `No SQL statements executed`。
  - （可选）将报表发送到指定的电子邮件地址。

要自动进行数据库访问审核与合规性调节：

1. 将以下各部分中的脚本之一复制到文件，然后将其保存到您的文件系统。将文件命名为 `parseActivityReport.ps1`（对于 Windows，请参阅“PowerShell 脚本 (parseActivityReport.ps1)”）或 `parseActivityReport.sh`（对于 Linux/UNIX，请参阅“Bash Shell 脚本 (parseActivityReport.sh)”）。
2. 仅限 **Windows**：通过将以下脚本复制到文件来创建名为 `parseActivityReport.bat` 的批处理文件。将文件保存在存储 `parseActivityReport.ps1` 的目录中。

```
@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. 修改 `parseActivityReport.bat` (Windows) 或 `parseActivityReport.sh` (Linux/UNIX) 以设置下表中参数的值。

表 3-11　要在脚本中包含的变量值

变量	说明
`epmuser`	服务管理员的用户名 示例： Windows：`set epmuser="jDoe"` Linux/UNIX：`epmuser="jDoe"`
`epmpassword`	服务管理员的密码或加密密码文件的位置有关创建加密密码文件的信息，请参阅 encrypt 命令。 如果密码中包含特殊字符，请参阅“处理特殊字符”。 示例： Windows：`set epmpassword = "Example"` Linux/UNIX：`epmpassword="Example"`
`epmurl`	Oracle Fusion Cloud Enterprise Performance Management 环境的 URL。 示例： Windows：`set epmurl="https://example.oraclecloud.com"` Linux/UNIX：`epmurl="https://example.oraclecloud.com"`
`report_email_to_address`	（可选）要将报表发送到的电子邮件地址。仅当指定此值时，才会通过电子邮件发送报表。 示例：**`john.doe@example.com`**

4. 仅适用于 **`parseActivityReport.sh`**：确保已为您的系统正确设置以下值：
   - *`JAVA_HOME`*
   - `epmautomatescript.sh` 的位置，通过更新 `epmautomatescript` 指令的值
5. 使用操作系统上提供的计划程序，将 `parseActivityReport.bat`（用于执行 `parseActivityReport.ps1`）或 `parseActivityReport.sh` 计划为每天运行一次。请参阅“自动执行脚本”。

PowerShell 脚本 (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 脚本 (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
```

复制用户和预定义角色分配

本节中的脚本有助于将一个环境的用户和预定义角色分配迁移到另一个环境。

关于脚本

使用两个不同的脚本：一个用于在身份域之间复制用户，另一个用于复制用户的预定义角色分配。这些脚本的运行顺序如下：

- 运行用于复制用户的脚本 (`replicateusers`) 并验证是否已在目标身份域中创建所有用户。运行此脚本的用户必须在这两个环境中都具有身份域管理员和服务管理员角色。
- 运行用于复制角色分配的脚本 (`replicatepredefinedroles`)。

> **注：**
>
> - 如果密码中包含特殊字符，请参阅“处理特殊字符”
> - 本节中的脚本仅适用于预定义角色：服务管理员、超级用户、用户和查看者。

运行脚本

有关创建必需的脚本和批处理文件的信息，请参阅以下主题：

- 将一个身份域的用户复制到另一个身份域
- 将预定义角色分配从一个环境复制到另一个环境

Windows 步骤

1. 创建 `replicateusers.bat`、`replicateusers.ps1`、`replicatepredefinedroles.bat` 和 `replicatepredefinedroles.ps1`，并将它们保存到您具有写入和执行权限的本地目录中。
2. 根据需要，使用源环境、目标环境和 Internet 代理服务器的信息更新批处理文件。
3. 运行用于执行 `replicateusers.ps1` 的 `replicateusers.bat`。必须将要分配给已复制用户的默认密码指定为命令行参数，如下所示：

```
replicateusers.bat Pwd_for_users
```

如果密码中包含特殊字符，请确保使用相应的转义符。请参阅“处理特殊字符”。

4. 运行 `replicatepredefinedroles.bat` 以创建与源环境中存在的角色分配相同的角色分配。

Linux/UNIX 步骤

1. 创建 `replicateusers.sh` 和 `replicatepredefinedroles.sh` 脚本，并将它们保存到您具有写入和执行权限的本地目录中。
2. 根据需要，使用源环境、目标环境和 Internet 代理服务器的信息更新 `replicateusers.sh` 和 `replicatepredefinedroles.sh`。
3. 运行 `replicateusers.sh`。必须将要分配给已复制用户的默认密码指定为命令行参数，如下所示：
   ```
   ./replicateusers.sh Pwd_for_users
   ```
   如果密码中包含特殊字符，请确保使用相应的转义符。请参阅“处理特殊字符”。
4. 运行 `replicatepredefinedroles.sh` 脚本以创建与源环境中存在的角色分配相同的角色分配。

将一个身份域的用户复制到另一个身份域

使用本节中的脚本可以将一个身份域的用户克隆到另一个身份域。运行这些脚本的用户必须在源环境和目标环境中具有身份域管理员和服务管理员角色。

Windows

通过复制本节中的脚本来创建 `replicateusers.bat` 和 `replicateusers.ps1`。

1. 通过复制以下脚本来创建 `replicateusers.ps1`：

```
# 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. 通过复制以下脚本来创建 `replicateusers.bat`：

```
@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. 更新 `replicateusers.bat`。有关必须指定的值，请参阅下表。

参数	说明
epmusersource	在源环境中具有身份域管理员和服务管理员角色的用户的用户名。 示例： Windows：`set epmusersource="jDoe"` Linux/UNIX：`epmusersource="jDoe"`
epmpwdsource	用户的密码或加密密码文件的绝对路径。 示例： Windows：`set epmpwdsource="Example"` Linux/UNIX：`epmpwdsource="Example"`
epmurlsource	要从中复制用户的环境的 URL。 示例： Windows：`set epmurlsource="https://example.oraclecloud.com"` Linux/UNIX：`epmurlsource="https://example.oraclecloud.com"`
epmidentitydomainsource	源环境使用的身份域的名称。 示例： Windows：`set epmidentitydomainsource="example_source_dom"` Linux/UNIX：`epmidentitydomainsource="example_source_dom"`
epmusertarget	在目标环境中具有身份域管理员和服务管理员角色的用户的用户名。 示例： Windows：`set epmusertarget="John.Doe"` Linux/UNIX：`set epmusertarget="John.Doe"`
epmpwdtarget	用户的密码或加密密码文件的绝对路径。 示例： Windows：`set epmpwdtarget="Example1"` Linux/UNIX：`epmpwdtarget="Example1"`
epmurltarget	要在其中创建用户的环境的 URL。 示例： Windows：`set epmurltarget="https://example.oraclecloud.com"` Linux/UNIX：`epmurltarget="https://example.oraclecloud.com"`
epmidentitydomaintarget	目标环境使用的身份域的名称。 示例： Windows：`set epmidentitydomaintarget="example_source_dom"` Linux/UNIX：`epmidentitydomaintarget="example_target_dom"`
proxyserverusername	使用具有 Internet 访问控制权的代理服务器对安全会话进行身份验证的用户名。删除此属性的所有未使用的实例。 示例： Windows：`set proxyserverusername="Example"` Linux/UNIX：`proxyserverusername="Example"`

参数	说明
proxyserverpassword	用于在代理服务器中对用户进行身份验证的密码。删除此属性的所有未使用的实例。 示例： Windows：set proxyserverpassword="examplePwd" Linux/UNIX：proxyserverpassword="examplePwd"
proxyserverdomain	为代理服务器定义的域的名称。删除此属性的所有未使用的实例。 示例： Windows：set proxyserverdomain="exampleDom" Linux/UNIX：proxyserverdomain="exampleDom"
emailtoaddress	（可选）要将角色分配报表发送到的电子邮件地址。仅当指定此值时，才会通过电子邮件发送报表。 示例：emailtoaddress=john.doe@example.com

Linux/UNIX

1. 通过复制以下脚本来创建 replicateusers.sh。

```
#!/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. 更新 `replicateusers.sh`。有关必须指定的值的信息，请参阅上表。此外，必须为以下属性指定值：
   - `javahome`：要在其中安装 Java 的目录的绝对路径。
   - `epmautomatescript`：`epmautomatescript.sh` 的位置；例如，`epmautomatescript="/home/user1/epmautomate/bin/epmautomate.sh"`

将预定义角色分配从一个环境复制到另一个环境

使用本节中的脚本可以将预定义角色分配从一个环境克隆到另一个环境。运行这些脚本的用户必须在这两个环境中都具有服务管理员角色。

> **注：**
>
> **如果使用的是本文档的 PDF 版本：**为避免出现换行和页脚信息从而导致这些脚本不可用，请从本主题的 HTML 版本复制脚本。

Windows

1. 通过复制以下脚本来创建 `replicatepredefineroles.ps1`。

```
# 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. 通过复制以下脚本来创建 `replicatepredefineroles.bat`。

```
@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. 根据需要更新 `replicatepredefineroles.bat`。有关必须为此文件中的属性设置的值的信息，请参阅下表。

更新 `replicatepredefineroles.bat`

参数	说明
`epmusersource`	在源环境中具有身份域管理员和服务管理员角色的用户的用户名。 **示例：** Windows：`set epmusersource="jDoe"` Linux/UNIX：`epmusersource="jDoe"`
`epmpwdsource`	用户的密码或加密密码文件的绝对路径。 **示例：** Windows：`set epmpwdsource="Example"` Linux/UNIX：`epmpwdsource="Example"`
`epmurlsource`	要从中复制用户的环境的 URL。 **示例：** Windows：`set epmurlsource="https://example.oraclecloud.com"` Linux/UNIX：`epmurlsource="https://example.oraclecloud.com"`
`epmidentitydomainsource`	源环境使用的身份域的名称。 **示例：** Windows：`set epmidentitydomainsource="example_source_dom"` Linux/UNIX：`epmidentitydomainsource="example_source_dom"`
`epmusertarget`	在目标环境中具有身份域管理员和服务管理员角色的用户的用户名。 **示例：** Windows：`set epmusertarget="John.Doe"` Linux/UNIX：`set epmusertarget="John.Doe"`
`epmpwdtarget`	用户的密码或加密密码文件的绝对路径。 **示例：** Windows：`set epmpwdtarget="Example1"` Linux/UNIX：`epmpwdtarget="Example1"`

参数	说明
epmurltarget	要在其中创建用户的环境的 URL。 **示例：** Windows：`set epmurltarget="https://example.oraclecloud.com"` Linux/UNIX：`epmurltarget="https://example.oraclecloud.com"`
epmidentitydomaintarget	目标环境使用的身份域的名称。 **示例：** Windows：`set epmidentitydomaintarget="example_target_dom"` Linux/UNIX：`epmidentitydomaintarget="example_target_dom"`
proxyserverusername	使用具有 Internet 访问控制权的代理服务器对安全会话进行身份验证的用户名。删除此属性的所有未使用的实例。 **示例：** Windows：`set proxyserverusername="Example"` Linux/UNIX：`proxyserverusername="Example"`
proxyserverpassword	用于在代理服务器中对用户进行身份验证的密码。删除此属性的所有未使用的实例。 **示例：** Windows：`set proxyserverpassword="examplePwd"` Linux/UNIX：`proxyserverpassword="examplePwd"`
proxyserverdomain	为代理服务器定义的域的名称。删除此属性的所有未使用的实例。 **示例：** Windows：`set proxyserverdomain="exampleDom"` Linux/UNIX：`proxyserverdomain="exampleDom"`
emailtoaddress	（可选）要将角色分配报表发送到的电子邮件地址。仅当指定此值时，才会通过电子邮件发送报表。 **示例：**`emailtoaddress=john.doe@example.com`

Linux/UNIX

1. 通过复制以下脚本来创建 `replicatepredefineroles.sh`。

```
#!/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. 更新 `replicatepredefineroles.sh`。有关必须指定的值的信息，请参阅上表。此外，必须为以下属性指定值：
   - `javahome`：要在其中安装 Java 的目录的绝对路径。
   - `epmautomatescript`：`epmautomatescript.sh` 的位置；例如，`epmautomatescript="/home/user1/epmautomate/bin/epmautomate.sh"`

创建每季度一次的云 EPM 升级节奏

使用这些脚本可创建用于跳过更新的自助服务解决方案，以便云 EPM 环境按季度更新，并实施为期两周的测试周期。在这种情况下，在更新测试环境两周后更新生产环境。

如果需要，此脚本还可以用于跳过每隔一月的更新。默认情况下，云 EPM 将每月更新应用于环境。您可以使用 skipUpdate 命令跳过向环境应用每月更新的过程或查看当前的跳过更新请求。可以使用本节包含的脚本自动运行 `skipUpdate` 命令。这些脚本将自动执行跳过更新过程，以便每季度或每隔一个月应用更新。

> **Note:**
>
> 1. 跳过更新的间隔不能超过连续两个月。例如，如果您尝试仅在二月、六月和十一月更新云 EPM 环境，脚本将引发错误。
> 2. 在间隔期间发生的所有更新都将在下次更新时应用于环境。例如，假设您使用此脚本将每季度更新安排在二月、五月、八月和十一月执行。在这种情况下，五月更新会将二月更新后发布的所有适用的云 EPM 每月更新和修补程序应用于您的环境。当应用更新时，维护过程可能会比平常花费更多时间。
> 3. 此脚本仅设置一个季度的更新节奏。请每月运行此脚本，以确保为全年配置更新节奏。

- Windows 脚本和说明
- UNIX/Linux 脚本和说明
- Groovy 脚本

重新运行该脚本

1. 要运行 Windows 和 Linux/UNIX 脚本：
   a. 创建 `input.properties` 文件，并使用环境信息进行相应的更新。将该文件保存到一个本地目录。此文件的内容因操作系统的不同而异。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
   b. 创建 `skip_update.ps1` (Windows PowerShell) 或 `skip_update.sh` (Linux/UNIX) bash 脚本，并将其保存到 `input.properties` 所在的目录。
   c. 启动脚本。

- Linux/UNIX：运行 `./skip_update.sh`。
- Windows PowerShell：运行 `skip_update.ps1`。

2. 要运行 Groovy 脚本，请在云 EPM 业务流程中使用 Groovy 屏幕，或使用 runBusinessRule 自动执行脚本。有关使用 EPM Automate 运行 Groovy 脚本的信息，请参阅“在不安装 EPM Automate 的情况下运行命令”。

Windows 脚本和说明

通过复制本节中的脚本，创建 `input.properties` 和 `skip_update.ps1`。

1. 通过复制以下脚本来创建 `input.properties`：

```
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updatemonths=02,05,08,11
```

2. 通过指定参数值更新 `input.properties`。

Table 3-12 input.properties 参数

参数	说明
`username`	服务管理员的用户名。
`password`	服务管理员的密码或加密密码文件的名称和位置。
`url`	更新节奏要设置为不是每月一次的环境的 URL。
`updatemonths`	应将 Oracle Fusion Cloud Enterprise Performance Management 更新应用于 `url` 参数所标识环境的月份的逗号分隔列表。例如，`updatemonths=02,05,08,11`。 月份必须指定为两位数字：01 表示一月，12 表示十二月。确保一月到九月的数字包含前面的零。该脚本尝试为没有包含在 `updatemonths` 参数值中的月份运行 skipUpdate 命令。例如，如果指定 `updatemonths=02,05,08,11`，则脚本尝试为一月、三月、四月、六月、七月、九月、十月和十二月设置跳过更新标志，以便更新仅在二月、五月、八月和十一月进行。

3. 通过复制以下脚本来创建 `skip_updates.ps1`：

```
# 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 脚本和说明

通过复制本节中的脚本，创建 input.properties 和 skip_update.sh。

1. 通过复制以下脚本来创建 input.properties：

```
javahome=JAVA_HOME
epmautomatescript=EPM_AUTOMATE_LOCATION
username=exampleAdmin
password=examplePassword.epw
url=exampleURL
updatemonths=02,05,08,11
```

2. 通过指定参数值更新 input.properties。

Table 3-13 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。
username	服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
url	更新节奏要设置为不是每月一次的环境的 URL。
updatemonths	应将 Oracle Fusion Cloud Enterprise Performance Management 更新应用于 url 参数所标识环境的月份的逗号分隔列表。例如，updatemonths=02,05,08,11。 月份必须指定为两位数字；一月到九月的数字包含前面的零。该脚本尝试为没有包含在 updatemonths 参数值中的月份运行 skipUpdate 命令。例如，如果指定 updatemonths=02,05,08,11，则脚本尝试为一月、三月、四月、六月、七月、九月、十月和十二月设置跳过更新标志，以便更新仅在二月、五月、八月和十一月进行。

3. 通过复制以下脚本来创建 skip_updates.sh：

```
#!/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 脚本

如果密码中包含特殊字符，请参阅“处理特殊字符”。此外，还要确保替换这些参数值以适应您的环境：

Table 3-14　要更改的参数

参数	说明
user	服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
url	更新节奏要设置为不是每月一次的环境的 URL。
updatemonths	应将 Oracle Fusion Cloud Enterprise Performance Management 更新应用于 `url` 参数所标识环境的月份的逗号分隔列表。例如，`updatemonths=02,05,08,11`。 月份必须指定为两位数字：01 表示一月，12 表示十二月。确保一月到九月的数字包含前面的零。该脚本尝试为没有包含在 `updatemonths` 参数值中的月份运行 skipUpdate 命令。例如，如果指定 `updatemonths=02,05,08,11`，则脚本尝试为一月、三月、四月、六月、七月、九月、十月和十二月设置跳过更新标志，以便更新仅在二月、五月、八月和十一月进行。

```
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')
```

创建每季度一次的云 EPM 升级节奏，并实施为期六周的测试周期

使用本节中的脚本可创建用于跳过更新的自助服务解决方案，以便 Oracle Fusion Cloud Enterprise Performance Management 环境按季度更新，并实施为期六周的测试周期。在这种情况下，在更新测试环境六周后更新生产环境。

默认情况下，云 EPM 将每月更新应用于环境。您可以使用 skipUpdate 命令跳过向环境应用每月更新的过程或查看当前的跳过更新请求。可以使用本节包含的脚本自动运行 `skipUpdate` 命令。这些脚本自动执行跳过更新过程，以便每季度应用更新，并实施为期六周的测试周期。

> **Note:**
>
> 1. 跳过更新的间隔不能超过连续三个月。如果您尝试仅在二月和十月更新云 EPM 环境，此脚本将引发错误。
> 2. 在间隔期间发生的所有更新都将在下次更新时应用于环境。例如，假设您使用此脚本将每季度更新仅安排在二月、五月、八月和十一月执行。在这种情况下，五月更新会将二月更新后发布的所有适用的云 EPM 每月更新和修补程序应用于您的环境。当应用更新时，维护过程可能会比平常花费更多时间。
> 3. 此脚本仅设置一个季度的更新节奏。
> 示例方案：测试环境更新周期确定为二月的第一个星期五（24.02 更新）、五月的第一个星期五（24.05 更新）、八月的第一个星期五（24.08 更新）和十一月的第一个星期五（24.11 更新）。将在三月的第三个星期五（24.02 更新）使用在二月的第一个星期五（24.02 更新）更新测试环境所用版本更新生产环境。将在六月的第三周（24.05 更新）、九月的第三周（24.08 更新）和十二月的第三周（24.11 更新）对生产环境进行类似更新。在这种情况下，生产环境不是更新到当前更新，而是测试环境当前采用的更新。

Windows 示例脚本

通过复制以下脚本来创建 `skip_update.ps1`。将其存储在本地目录中。有关运行以下脚本的信息，请参阅“重新运行该脚本”：

```
# 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 示例脚本

通过复制以下脚本来创建 `skip_update.sh`。将其存储在本地目录中。有关运行以下脚本的信息，请参阅“重新运行该脚本”：

```
#!/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
```

Groovy 脚本

通过复制以下脚本并对其进行更新来创建 `skip_update.groovy` Groovy 脚本。有关运行以下脚本的信息，请参阅“重新运行该脚本”：
更新此 Groovy 脚本中的以下变量：

- `username`：更新节奏要设置为不是每月一次的环境的服务管理员的用户名。
- `password`：服务管理员的密码或加密密码文件的名称和位置。
- `url`：更新节奏要设置为不是每月一次的环境的 URL。
- `updateversions`：应该应用于 `url` 参数标识的环境的云 EPM 更新列表（以逗号分隔）。例如 `updateversions=02,05,08,11`。
  版本必须指定为两位数字；更新 01 到 09 包含前面的零。该脚本尝试为没有包含在 `updateversions` 参数值中的更新运行 skipUpdate 命令。例如，如果指定 `updateversions=02,05,08,11`，则脚本尝试针对 01（一月）、03（三月）、04（四月）、06（六月）、07（七月）、09（九月）、10（十月）和 12（十二月）更新设置跳过更新标志。在这种情况下，云 EPM 更新 02（二月）、05（五月）、08（八月）和 11（十一月）将应用于环境。
- `podtype`：云 EPM 环境类型。有效值为 `test` 和 `prod`。
- `proxyserverusername`：用于在控制 Internet 访问的代理服务器中对安全会话进行身份验证的用户名。
- `proxyserverpassword`：用于在代理服务器中对用户进行身份验证的密码。
- `proxyserverdomain`：为代理服务器定义的域的名称。

> **Note:**
>
> 如果不使用代理服务器，则不要为 `proxyserverusername`、`proxyserverpassword` 和 `proxyserverdomain` 参数指定任何值。

```
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')
```

创建 input.properties 文件以运行 skip_update Windows 和 Linux/UNIX 脚本

要运行 skip_update.ps1 或 skip_update.sh，请创建 input.properties 文件并使用环境信息进行相应的更新。将该文件保存到一个本地目录。此文件的内容因操作系统的不同而异。

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
updateversions=01,04,07,10
podtype=test
```

Table 3-15 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
url	更新节奏要设置为不是每月一次的环境的 URL。
updateversions	updateversions：应该应用于 url 参数标识的环境的云 EPM 更新列表（以逗号分隔）。例如 updateversions=02,05,08,11。版本必须指定为两位数字；更新 01 到 09 包含前面的零。该脚本尝试为没有包含在 updateversions 参数值中的更新运行 skipUpdate 命令。例如，如果指定 updateversions=02,05,08,11，则脚本尝试针对 01（一月）、03（三月）、04（四月）、06（六月）、07（七月）、09（九月）、10（十月）和 12（十二月）更新设置跳过更新标志。在这种情况下，云 EPM 更新 02（二月）、05（五月）、08（八月）和 11（十一月）将应用于环境。
podtype	云 EPM 环境类型。有效值为 test 和 prod。

重新运行该脚本

1. 仅限 **Windows 和 Linux/UNIX**：
   - 通过复制上一节的脚本来创建 `skip_update.ps1` 或 `skip_update.sh`。
   - 创建 `input.properties` 文件，并将其保存在 `skip_update` 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件以运行 skip_update Windows 和 Linux/UNIX 脚本”。
     请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
   - 启动脚本。
     - **Windows PowerShell**：运行 `skip_update.ps1`。
     - **Linux/UNIX**：运行 `./skip_update.sh`。
2. **Groovy**：
   - 创建 `skip_update.groovy` Groovy 脚本并根据需要对其进行更新。
   - 在云 EPM 业务流程中使用 Groovy 屏幕，或使用 runBusinessRule 自动执行脚本。有关使用 EPM Automate 运行 Groovy 脚本的信息，请参阅“在不安装 EPM Automate 的情况下运行命令”。

Planning、Consolidation、Tax Reporting 和 Enterprise Profitability and Cost Management 的示例方案

本节中提供的脚本可帮助您自动执行 Planning（包括 Planning 模块）、Financial Consolidation and Close、Tax Reporting 和 Enterprise Profitability and Cost Management 环境中的任务。

另请参阅：

- 自动从聚合存储多维数据集导出大量单元格
  使用本节中的 PowerShell 或 Bash 脚本可从聚合存储 (ASO) 多维数据集导出大量单元格。
- 将元数据导入应用程序
  使用这些脚本可从文件手动导入应用程序元数据。
- 导入数据、运行计算脚本并将数据从块存储数据库复制到聚合存储数据库
  使用这些脚本可从文件导入数据，刷新多维数据集，运行对多维数据集进行计算的业务规则，然后将数据推送到 ASO 多维数据集。
- 导出和下载元数据和数据
  使用这些脚本导出应用程序元数据和数据，然后将导出文件下载到本地目录。
- 导出和下载应用程序数据
  使用这些脚本可导出应用程序数据，然后将其下载到本地目录。
- 自动存档应用程序审核记录
  使用本节中的 Windows 和 Linux 脚本可自动将应用程序审核数据导出并存档到本地计算机。
- 将数据文件上传到环境并运行数据加载规则
  使用这些脚本可将文件上传到环境，然后运行数据规则将数据从文件导入到应用程序中。
- 自动执行每日数据集成
  此方案介绍了如何使用示例脚本定期自动整合数据。

自动从聚合存储多维数据集导出大量单元格

使用本节中的 PowerShell 或 Bash 脚本可从聚合存储 (ASO) 多维数据集导出大量单元格。

由于 Oracle Essbase `QUERYRESULTLIMIT` 施加的限制，无法从用户界面导出大量数据。本节中提供的 PowerShell 脚本会将导出操作拆分为指定数量的作业、运行每个作业、下载导出的数据，以及将导出文件连接到一个导出文件，以确保仅存在一个标题。

> **注：**
>
> 这些脚本将执行类型为导出数据的现有作业。有关创建作业的详细说明，请参阅《管理 *Planning*》中的“管理作业”。

PowerShell 脚本

```
$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 脚本

```
#!/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"
```

要从聚合存储 (ASO) 多维数据集导出大量单元格：

1. 复制 PowerShell 或 Bash 脚本，并将脚本（例如，`ASOCellExport.ps1` 或 `ASOCellExport.sh`）保存到您的文件系统。
2. 修改脚本文件并设置参数值。有关详细信息，请参阅下表。

表 3-16　要包含在 PowerShell 和 Bash 脚本中的变量值

变量	说明
`user`	服务管理员的域名和用户名，采用 *DOMAIN.USER* 格式。 **示例：** **Windows：** `$user = 'exampleDomain.jDoe'` **Linux/UNIX：** `user = 'exampleDomain.jDoe'`
`pass`	服务管理员的密码或加密密码文件的位置有关创建加密密码文件的信息，请参阅 encrypt 命令。 **示例：** **Windows：** `$pass = 'Example'` **Linux/UNIX：** `pass = 'Example'`
`serverURL`	Oracle Fusion Cloud Enterprise Performance Management 环境的 URL。 **示例：** **Windows：** `$serverURL = 'https://example .oraclecloud.com'` **Linux/UNIX：** `serverURL = 'https://example .oraclecloud.com'`
`applicationName`	Planning、Financial Consolidation and Close、Tax Reporting 或 Enterprise Profitability and Cost Management 应用程序的名称。 **示例：** **Windows：** `$applicationName = 'Vision'` **Linux/UNIX：** `applicationName = 'Vision'`

表 3-16 （续） 要包含在 PowerShell 和 Bash 脚本中的变量值

变量	说明
cubeName	应用程序中多维数据集的名称。 示例： **Windows：** `$cubeName = 'VisASO'` **Linux/UNIX：** `cubeName = 'VisASO'`
splitDimension	维的名称，该维的成员用于将导出拆分为几个组。 示例： **Windows：** `$splitDimension = 'Account'` **Linux/UNIX：** `splitDimension = 'Account'`
topLevelMemberForExport	维子层次成员的名称，将在其下创建 0 级成员列表。 示例： **Windows：** `$topLevelMemberForExport = 'Total Cash Flow'` **Linux/UNIX：** `topLevelMemberForExport = 'Total Cash Flow'`
exportJobName	类型为导出数据的现有作业的名称。在此作业中指定的设置将被您在脚本中设置的参数覆盖。 示例： **Windows：** `$exportJobName = 'ASO Cell Export'` **Linux/UNIX：** `exportJobName = 'ASO Cell Export'`
exportFilePrefix	唯一标识由导出作业生成的文件的文件名前缀。 示例： **Windows：** `$exportFilePrefix = 'cashflow'` **Linux/UNIX：** `exportFilePrefix = 'cashflow'`
columnMembers	要在导出中包含的成员列。 示例： **Windows：** `$columnMembers = 'Period'` **Linux/UNIX：** `columnMembers = 'Period'`
povMembers	要在导出中包含的视点。POV 成员必须包含所有其他维，并且可以包含如下所示的函数： `ILvl0Descendants(YearTotal), ILvl0Descendants(Year), ILvl0Descendants(Scenario), ILvl0Descendants(Version), ILvl0Descendants(P_TP), ILvl0Descendants(AltYear)` 示例： **Windows：** `$povMembers = 'YTD'` **Linux/UNIX：** `povMembers = 'YTD'`
numberOfExportFiles	要为此导出操作执行的作业数。如果导出由于查询限制而仍然失败，请增加此数字。 示例： **Windows：** `$numberOfExportFiles = 3` **Linux/UNIX：** `numberOfExportFiles = 3`

3. 使用 Windows 计划程序或 cron 作业，安排脚本在方便的时间执行。有关详细步骤，请参阅“自动执行脚本”。

将元数据导入应用程序

使用这些脚本可从文件手动导入应用程序元数据。

这些脚本执行以下活动：

- 登录环境。
- 上传元数据文件。
- 使用作业将元数据从上传的文件导入到应用程序中。
- 刷新多维数据集。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建 `importMetadata.ps1`。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建 `importMetadata.sh`。将其存储在本地目录中。

```
#!/bin/bash
. ./input.properties
export JAVA_HOME=${javahome}
${epmautomatescript} login "${username}" "${passwordfile}" "${serviceURL}"
${epmautomatescript} uploadfile "${file1}"
${epmautomatescript} importmetadata "${jobName}" "${file1}"
${epmautomatescript} refreshcube
${epmautomatescript} logout
```

创建 input.properties 文件

通过复制以下内容之一并使用环境信息进行相应的更新来创建 `input.properties` 文件。将文件保存在存储 `importMetadata.ps1` 或 `importMetadata.sh` 的目录中。

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
```

表 3-17 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
File1	包含要导入的元数据的 ZIP 文件的名称。
JobName	用于导入元数据的作业。

运行脚本

1. 通过复制上一节的脚本来创建 importMetadata.ps1 或 importMetadata.sh。
2. 创建 input.properties 文件，并将其保存在 importMetadata 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 importMetadata.ps1。
   - **Linux/UNIX**：运行 ./importMetadata.sh。

导入数据、运行计算脚本并将数据从块存储数据库复制到聚合存储数据库

使用这些脚本可从文件导入数据，刷新多维数据集，运行对多维数据集进行计算的业务规则，然后将数据推送到 ASO 多维数据集。

这些脚本执行以下操作：

- 登录环境。
- 上传文件 data.csv。
- 使用作业 loadingq1data 将数据从 data.csv 导入到应用程序中。

- 刷新多维数据集。
- 运行用于转换数据的业务规则。
- 使用作业将数据推送到聚合存储数据库。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建 `importDataPlus.ps1`。将其保存到本地目录。

```
$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 示例脚本

通过复制以下脚本来创建 `importDataPlus.ps1`。将其保存到本地目录。

```
#!/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
```

创建 input.properties 文件

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
```

表 3-18 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
File1	要将数据从其加载到应用程序中的导入文件。
iimportDataJobName	用于导入数据的作业的名称。
businessRuleName	要对导入的数据运行的业务规则
planTypeMapName	用于将数据从 BSO 数据库复制到 ASO 数据库或从 BSO 数据库复制到另一个 BSO 数据库的规划类型映射。
param1Key	要运行业务规则的运行时提示 1。

表 3-18 （续） input.properties 参数

参数	说明
param1Value	运行时提示 1 的值。
param2Key	要运行业务规则的运行时提示 2。
param2Value	运行时提示 2 的值。
clearData	指示是否要删除接收数据库中的数据。指定 false 将保留数据。

运行脚本

1. 通过复制上一节的脚本来创建 importDataPlus.ps1 或 importDataPlus.sh。
2. 创建 input.properties 文件，并将其保存在 importDataPlus 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 importDataPlus.ps1。
   - **Linux/UNIX**：运行 ./importDataPlus.sh。

导出和下载元数据和数据

使用这些脚本导出应用程序元数据和数据，然后将导出文件下载到本地目录。

这些脚本完成以下活动：

- 登录环境。
- 使用指定作业将元数据导出到 zip 文件中。
- 使用指定作业将应用程序数据导出到 zip 文件中。
- 列出收件箱/发件箱的内容。
- 将导出的数据文件下载到本地计算机。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建 exportDownloadMetadataAndData.ps1。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建 `exportDownloadMetadataAndData.sh`。将其存储在本地目录中。

```
#!/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
```

创建属性文件

通过复制以下内容之一并使用环境信息进行相应的更新来创建 `input.properties` 文件。将文件保存在存储 `exportDownloadMetadataAndData.ps1` 或 `exportDownloadMetadataAndData.sh` 的目录中。

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
```

表 3-19 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。

表 3-19 （续） input.properties 参数

参数	说明
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
exportFile1	要将元数据导出到的文件的名称。
exportFile2	要将数据导出到的文件的名称。
exportDataJobName1	用于导出元数据的作业。
exportDataJobName2	用于导出数据的作业。

运行脚本

1. 通过复制上一节的脚本来创建 exportDownloadMetadataAndData.ps1 或 exportDownloadMetadataAndData.sh。
2. 创建 input.properties 文件，并将其保存在 exportDownloadMetadataAndData 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅"创建属性文件"。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 exportDownloadMetadataAndData.ps1。
   - **Linux/UNIX**：运行 ./exportDownloadMetadataAndData.sh。

导出和下载应用程序数据

使用这些脚本可导出应用程序数据，然后将其下载到本地目录。

这些脚本执行以下操作：

- 登录环境。
- 使用您指定的作业备份两组数据。
- 下载导出的数据文件。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建 exportDownloadData.ps1。将其保存到本地目录。

```
$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 示例脚本

通过复制以下脚本来创建 `exportDownloadData.sh`。将其保存到本地目录。

```
#!/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
```

创建 input.properties 文件

通过复制以下内容之一并使用环境信息进行相应的更新来创建 `input.properties` 文件。将文件保存在存储 `exportDownloadData.ps1` 或 `exportDownloadData.sh` 的目录中。

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
```

表 3-20　input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
exportFile1 和 exportFile2	要将数据导出到的文件的名称。
exportDataJobName1 和 exportDataJobName2	用于导出数据的作业。

运行脚本

1. 通过复制上一节的脚本来创建 exportDownloadData.ps1 或 exportDownloadData.sh。
2. 创建 input.properties 文件，并将其保存在 exportDownloadData 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“表 1”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 exportDownloadData.ps1。
   - **Linux/UNIX**：运行 ./exportDownloadData.sh。

自动存档应用程序审核记录

使用本节中的 Windows 和 Linux 脚本可自动将应用程序审核数据导出并存档到本地计算机。

应用程序审核数据只保留 365 天。要防止超过 365 天的历史审核数据丢失，请自定义这些脚本，然后每 180 天执行一次，或按您的数据保留策略所需的频率执行。

> **Note:**
>
> 这些脚本进行了定制，会将数据存档在本地存储。您可以修改脚本，以将导出的审核数据文件存档到网络存储或存储云（例如 Oracle Object Storage）中。

Table 3-21　参数及其值

参数	值
url	环境的 URL。 示例： • **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

Table 3-21 (Cont.) 参数及其值

参数	值
user	要登录环境下载审核数据的服务管理员的用户名。 示例： • **Windows**：set user=ExampleAdmin • **Linux**：user=ExampleAdmin
password	服务管理员的密码（不建议使用）或加密密码文件的名称和位置。有关创建加密密码文件的信息，请参阅 encrypt 命令。 示例： • **Windows**：set password="C:\mySecuredir\password.epw" • **Linux**：password="/home/user1/mySecuredir/password.epw"
AuditFileName	审核数据文件的名称。为了使此文件唯一，脚本在此文件名后附加审核数据导出时间戳。 示例： • **Windows**：set AuditFileName=AuditData • **Linux**：AuditFileName=AuditData
NumberOfBackups	存储中保留的备份文件数。默认值为 10，超出后将根据需要替换最早的备份。 示例： • **Windows**：set NumberOfBackups=20 • **Linux**：NumberOfBackups=20
仅限 Linux 脚本	
epmautomatescript	安装 EPM Automate 的位置。 示例：/home/user1/epmautomate/bin/epmautomate.sh
javahome	*JAVA_HOME* 位置。 示例：/home/user1/jdk1.8.0_191/

Windows 脚本

创建一个批处理文件（例如 AuditExport.bat），在其中包含类似以下内容的脚本，以自动将审核数据导出并下载到本地计算机。

```
@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 脚本

创建一个 shell 脚本（例如 `AuditExport.sh`），在其中包含类似以下内容的脚本，以自动将审核数据导出并下载到本地计算机。

```
#!/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 --
```

将数据文件上传到环境并运行数据加载规则

使用这些脚本可将文件上传到环境，然后运行数据规则将数据从文件导入到应用程序中。

先决条件

- 数据管理中存在以下定义：
  - 名为 `VisionActual`。 的数据加载规则定义假设该数据规则未指定输入文件的文件路径。
  - `Mar-15` 至 `Jun-15` 的期间定义
- 包含数据且格式正确的数据文件 (`GLActual.dat`)。

要导入数据并运行数据加载规则，需运行命令来完成以下步骤：

- 登录环境。
- 将包含 `Mar-15` 至 `Jun-15` 期间数据的文件 `GLActual.dat` 上传到数据管理文件夹 `inbox/Vision` 中。
- 使用数据加载规则 `VisionActual`、起始期间 `Mar-15`、结束期间 `Jun-15` 以及导入模式 `REPLACE` 将数据从 `GLActual.dat` 导入到数据管理中。
- 使用 `STORE_DATA` 选项导出数据，以将数据管理临时表中的数据和现有应用程序数据合并。
- 注销。

Windows 示例脚本

通过复制以下脚本来创建 `runDataLoadRule.ps1`。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建 `runDataLoadRule.sh`。将其存储在本地目录中。

```
#!/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
```

创建 input.properties 文件

通过复制以下内容之一并使用环境信息进行相应的更新来创建 `input.properties` 文件。将文件保存在存储 `runDataLoadRule.ps1` 或 `runDataLoadRule.sh` 的目录中。

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
```

表 3-22 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
dataFile	包含要使用数据规则导入的数据的文件。
dataFileUploadLocation	要向其上传数据文件的位置。
dataRuleName	在数据集成中定义的数据加载规则的名称。
startPeriod	要加载数据的第一个期间。此期间名称必须在数据集成期间映射中进行定义。
endPeriod	对于多期间数据加载，这是要加载数据的最后一个期间。对于单期间加载，使用与起始期间相同的期间。此期间名称必须在数据集成期间映射中进行定义。
importMode	将数据导入到数据集成中时所用的模式。使用 APPEND、REPLACE 或 RECALCULATE。使用 NONE 将跳过将数据导入到临时表的操作。
exportMode	将数据导出到应用程序时所用的模式。使用数据集成。使用 STORE_DATA、ADD_DATA、SUBTRACT_DATA 或 REPLACE_DATA。使用 NONE 将跳过将数据从数据集成导出到应用程序的操作。 **注：** Financial Consolidation and Close 仅支持 MERGE 和 NONE 模式。

重新运行该脚本

1. 通过复制上一节的脚本来创建 runDataLoadRule.ps1 或 runDataLoadRule.sh。
2. 创建 input.properties 文件，并将其保存在 runDataLoadRule 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 runDataLoadRule.ps1。
   - **Linux/UNIX**：运行 ./runDataLoadRule.sh。

自动执行每日数据集成

此方案介绍了如何使用示例脚本定期自动整合数据。

创建包含类似以下脚本的批处理 (.bat) 或 shell (.sh) 文件以自动执行与数据集成相关的活动。适用于 Windows 的以下示例脚本通过完成以下活动可以每日自动执行应用程序数据集成：

- 登录环境。
- 删除 DailyPlanData（如果存在）。
- 将 DailyPlanData 上传到服务实例。
- 对规划类型 Plan1 运行业务规则 Clear Plan Targets。
- 使用作业名称 LoadDailyPlan 导入数据。
- 运行业务规则 Balance Sheet - Plan。
- 运行业务规则 Allocate Plan Targets。
- 删除 DailyTarget.zip（如果存在）。
- 使用作业名称 ExportDailyTarget 将数据导出到 DailyTarget.zip。
- 将 DailyTarget.zip 下载到服务器并附加时间戳。
- 从环境中注销。

> **注：**
>
> 如果将该脚本改作他用，请确保修改 SET url 和 SET user 参数的值。此外，您可能要修改 dataimportfilename、dataexportfilename、importdatajobname、exportdatajobname、br_clear、br_calculatebalancesheet 和 br_allocatetarget 参数的值来满足需求

有关使用 Windows 任务计划程序计划脚本的信息，请参阅“自动执行脚本”。

```
@echo off

rem Sample Script to demonstrate daily data integration with
rem Cloud EPM 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%
```

Account Reconciliation 示例方案

另请参阅：

- 将预先设置格式的余额加载到期间内
  使用这些脚本可将所上传文件中的映射数据导入到 Account Reconciliation 环境中。
- 上传和导入备份快照
  使用这些脚本可将备份快照上传并导入到 Account Reconciliation 环境中。
- 存档旧的匹配事务和清除存档的事务
  使用本节中的脚本可存档等于或超过指定天数的匹配事务（包括支持和调整详细信息），然后从 Account Reconciliation 中清除存档的事务。存档的匹配事务记录在 ZIP 文件中。

将预先设置格式的余额加载到期间内

使用这些脚本可将所上传文件中的映射数据导入到 Account Reconciliation 环境中。

Windows 示例脚本

通过复制以下脚本来创建一个名为 `runPreformattedBalances.ps1` 的文件。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建一个名为 `runPreformattedBalances.sh` 的文件。将其存储在本地目录中。

```
#!/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
```

示例 input.properties 文件

要运行 `runPreformattedBalances` 脚本，请创建 `input.properties` 文件并使用环境信息进行相应的更新。将文件保存在存储 `runPreformattedBalances.sh` 或 `runPreformattedBalances.ps1` 的目录中。

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
```

表 3-23 input.properties 参数

参数	说明
`javahome`	`JAVA_HOME` 位置。仅限 Linux/UNIX。
`epmautomatescript`	EPM Automate 可执行文件 (`epmautomate.sh`) 的绝对路径。仅限 Linux/UNIX。
`username`	服务管理员的用户名。
`password`	服务管理员的密码或加密密码文件的名称和位置。
`serviceURL`	托管应用程序的环境的 URL，您要将预先设置格式的余额加载到此应用程序中。
`dataFile`	包含预先设置格式的余额的 CSV 文件（通常是基于总帐创建的），您要将这些余额加载到应用程序中。必须已经使用 uploadFile 命令将此文件上传到环境。
`period`	向其上传预先设置格式的余额的调节期间。
`balanceType`	`dataFile` 中包含的预先设置格式的余额的类型。

表 3-23 （续） input.properties 参数

参数	说明
currencyBucket	预先设置格式的余额的货币组。

重新运行该脚本

1. 通过复制上一节的脚本来创建 runPreformattedBalances.ps1 或 runPreformattedBalances.sh。
2. 仅限 **Windows 和 Linux/UNIX**：
   - 创建 input.properties 文件，并将其保存在 runPreformattedBalances 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“表 1”。
     请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
   - 启动脚本。
     - **Windows PowerShell**：运行 runPreformattedBalances.ps1。
     - **Linux/UNIX**：运行 ./runPreformattedBalances.sh。

上传和导入备份快照

使用这些脚本可将备份快照上传并导入到 Account Reconciliation 环境中。

Windows 示例脚本

通过复制以下脚本来创建一个名为 importBackupSnapshot.ps1 的文件。将其存储在本地目录中。

```
$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 示例脚本

通过复制以下脚本来创建一个名为 importBackupSnapshot.sh 的文件。将其存储在本地目录中

```
#!/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
```

示例 input.properties 文件

要运行 `importBackupSnapshot` 脚本，请创建 `input.properties` 文件并使用环境信息进行相应的更新。将文件保存在存储 `importBackupSnapshot.sh` 或 `importBackupSnapshot.ps1` 的目录中。

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
```

表 3-24 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要导入快照的环境的 URL。
snapshotName	要从其导入对象和数据的快照的名称。必须已经使用 uploadFile 命令将此快照上传到环境。
userPassword	必须为因导入快照而在身份域中创建的所有新用户分配的默认密码。

重新运行该脚本

1. 通过复制上一节的脚本来创建 `importBackupSnapshot.ps1` 或 `importBackupSnapshot.sh`。
2. 创建 `input.properties` 文件，并将其保存在 `runPreformattedBalances` 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“示例 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。

- **Windows PowerShell**：运行 `importBackupSnapshot.ps1`。
- **Linux/UNIX**：运行 `./importBackupSnapshot.sh`。

存档旧的匹配事务和清除存档的事务

使用本节中的脚本可存档等于或超过指定天数的匹配事务（包括支持和调整详细信息），然后从 Account Reconciliation 中清除存档的事务。存档的匹配事务记录在 ZIP 文件中。

脚本的工作原理

1. 使用 `input.properties` 文件中的信息登录环境
2. 运行以下 `archiveTmTransactions` 命令来创建存档。生成的 ZIP 文件和日志文件分别使用默认名称 `Archive_Transactions_INTERCO_JOB_ID.zip` 和 `Archive_Transactions_INTERCO_JOB_ID.log`

   ```
   epmautomate archiveTmTransactions INTERCO 365 filterOperator=contains
   filterValue=14001
   ```

   您可以通过修改 `input.properties` 文件来更改命令参数。
3. 将日志文件和包含存档事务的 .ZIP 文件下载到本地计算机。如果未找到匹配的事务，则脚本将显示错误消息。
4. 将包含存档事务的 .ZIP 文件复制到 Oracle Object Store。
5. 运行 `purgeArchivedTmTransactions` 命令（使用 archiveTmTransactions 作业的作业 ID）从应用程序中删除存档的匹配事务。

重新运行该脚本

1. 创建 `input.properties` 文件，并使用环境信息进行相应的更新。将该文件保存到一个本地目录。在本次讨论中，此目录名为 `parentsnapshotdirectory`。此文件的内容因操作系统的不同而异。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
2. 创建 `transaction_match.ps1` (Windows PowerShell) 或 `transaction_match.sh` (Linux/UNIX) 脚本，并将其保存到 `input.properties` 所在的 `parentsnapshotdirectory` 中。
3. 启动脚本。
   - Linux/UNIX：运行 `./transaction_match.sh`。
   - Windows PowerShell：运行 `transaction_match.ps1`。

创建 input.properties 脚本

通过复制并更新以下脚本来创建 `input.properties`。

```
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
```

> **注：**
>
> 仅限 **Windows**：从 `input.properties` 文件中删除以下属性：
>
> - `javahome=JAVA_HOME`
> - `epmautomatescript=EPM_AUTOMATE_LOCATION`
>
> 如果没有为 Windows 网络环境启用在代理服务器中进行身份验证，则从 `input.properties` 文件中删除以下属性。
>
> - `proxyserverusername`
> - `proxyserverpassword`
> - `proxyserverdomain`

表 3-25　input.properties 参数

参数	说明
`javahome`	EPM Automate 使用的 JDK 的安装目录。从 Windows 版本的 `input.properties` 中删除此条目。 **示例：** `javahome=./home/JDK/bin`
`epmautomatescript`	安装 EPM Automate 的目录。从 Windows 版本的 `input.properties` 中删除此条目。 **示例：** `epmautomatescript=./home/utils/EPMAutomate/bin`
`epmusername`	服务管理员、超级用户、用户或有权存档匹配事务的查看者的名称。 **示例：** `epmusername=ServiceAdmin`
`epmuserpassword`	标识为 `epmusername` 的用户的加密密码文件。 **示例：** `epmpassword=myPwd.epw`
`epmurl`	要存档匹配事务的环境的 URL。 **示例：** `epmurl=https://test-cloudpln.pbcs.us1.oraclecloud.com`
`objectstorageusername`	具有向 Oracle Object Storage Cloud 写入所需访问权限的用户的 ID。对于在联合身份提供程序中创建的用户，请指定用户的全限定名称（例如 `exampleIdP/jdoe` 或 `exampleIdP/john.doe@example.com`，其中 `exampleIdP` 是联合身份提供程序的名称）。对于其他用户，请指定用户 ID。 **示例：** `epmusername=myIdP/jdoe`
`objectstoragepassword`	与 `objectstorageusername` 中标识的用户关联的 Swift 密码或身份验证令牌。此密码与用户用于登录对象存储控制台的密码不同。身份验证令牌是 Oracle 生成的令牌，您可以将其与第三方 API 一起用于身份验证，例如与 Swift 客户端一起用于身份验证。有关创建此令牌的说明，请参阅 *Oracle Cloud Infrastructure* 文档中的“创建身份验证令牌”。 **示例：** `objectstoragepassword=jDoe_PWD`

表 3-25 （续） input.properties 参数

参数	说明
objectstorageurl	Oracle Object Storage Cloud 存储桶的 URL，后面可以附加对象名称。 **示例：** objectstorageurl=https://swiftobjectstorage.region_identifier.oraclecloud.com/v1/namespace/MT_Archives/2023_archives
matchtype	应存档匹配事务的匹配类型的标识符 (TextID)。 **示例：** matchtype=cashrecon
age	自匹配事务以来的天数。将存档超过或等于此值的匹配事务。 **示例：** age=180
filteroperator	用于标识包含要存档的匹配事务的帐户的筛选条件。必须是以下项之一：equals、not_equals、starts_with、ends_with、contains 或 not_contains。 此值与 filterValue 相结合，用于标识要存档匹配事务的帐户。 **示例：** filteroperator=not_equals
filtervalues	一个或多个用于标识要存档的事务的筛选器值。如果 equals 或 not_equals 指定为 filterOperator，可以使用空格分隔的列表指定多个值。如果指定多个值，将选择与任何筛选器运算符和筛选器值组合匹配的帐户中的事务进行存档。 **示例：** filterValue=101-120 filterValue=140-202
proxyserverusername	使用具有 Internet 访问控制权的代理服务器对安全会话进行身份验证的用户名。 **示例：** proxyserverusername=myProxyserver
proxyserverpassword	用于在代理服务器中对用户进行身份验证的密码。 **示例：** proxyserverpassword=myProxyserver_pwd
proxyserverdomain	为代理服务器定义的域的名称。 **示例：** proxyserverdomain=myProxyDomain

Windows 脚本

通过复制以下脚本来创建 transaction_match.ps1。将其存储在存储 input.properties 的文件夹中。

```
# 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 脚本

通过复制以下脚本来创建 `transaction_match.sh`。将其存储在存储 `input.properties` 的文件夹中。

```
#!/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."
```

Profitability and Cost Management 示例方案

这些方案介绍了如何使用 EPM Automate 命令执行某些常见的 Profitability and Cost Management 任务。

另请参阅：

- 将元数据导入应用程序中
  使用这些脚本可上传元数据文件并将维元数据从其导入到 Profitability and Cost Management 应用程序中。
- 导入数据并运行程序规则
  使用这些脚本可上传数据文件并将数据从上传的文件导入到 Profitability and Cost Management 业务流程中。

将元数据导入应用程序中

使用这些脚本可上传元数据文件并将维元数据从其导入到 Profitability and Cost Management 应用程序中。

这些脚本执行以下操作：

- 登录环境。
- 上传元数据文件。
- 将元数据从上传的文件导入到应用程序中。
- 启用应用程序。
- 注销。

Windows 脚本

通过复制以下脚本来创建 `importMetadata.ps1`。

```
$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 脚本

通过复制以下脚本来创建 `importMetadata.sh`。

```
#!/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
```

创建 input.properties 文件

要运行 importMetadata 脚本，请创建 input.properties 文件并使用环境信息进行相应的更新。将文件保存在存储 importMetadata.ps1 或 importMetadata.sh 的目录中。

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
```

表 3-26 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
applicationName	要向其加载数据的 Profitability and Cost Management 的名称。
dataFileName	包含要导入的元数据的文件的名称。
dataFileNameDestination	元数据文件的上传位置。

运行脚本

1. 通过复制上一节的脚本来创建 importMetadata.ps1 或 importMetadata.sh。
2. 创建 input.properties 文件，并将其保存在 importMetadata 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。

- **Windows PowerShell**：运行 `importMetadata.ps1`。
- **Linux/UNIX**：运行 `./importMetadata.sh`。

导入数据并运行程序规则

使用这些脚本可上传数据文件并将数据从上传的文件导入到 Profitability and Cost Management 业务流程中。

这些脚本完成以下步骤：

- 登录环境。
- 上传包含要加载的数据的数据文件。
- 上传包含数据规则的规则文件。
- 将数据从数据文件加载到应用程序中以覆盖现有值。
- 运行规则文件中的所有规则。
- 注销。

Windows 脚本

通过复制以下脚本来创建一个名为 `importData.ps1` 的文件。将其存储在本地目录中。

```
$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 脚本

通过复制以下脚本来创建一个名为 `importData.sh` 的文件。将其存储在本地目录中。

```
#!/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
```

创建 input.properties 文件

要运行 importData 脚本，请创建 input.properties 文件并使用环境信息进行相应的更新。将文件保存在存储 importData.ps1 或 importData.sh 的目录中。

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
```

表 3-27 input.properties 参数

参数	说明
javahome	JAVA_HOME 位置。仅限 Linux/UNIX。
epmautomatescript	EPM Automate 可执行文件 (epmautomate.sh) 的绝对路径。仅限 Linux/UNIX。
username	同时具有身份域管理员角色的服务管理员的用户名。
password	服务管理员的密码或加密密码文件的名称和位置。
serviceURL	要从其生成快照的环境的 URL。
applicationName	要向其加载数据的 Profitability and Cost Management 的名称。
dataFileName	包含要导入的数据的文件的名称。
rulesFileName	包含要对导入数据运行的规则的文件的名称。
fileDestination	要向其上传数据和规则文件的位置。
clearAllDataFlag	指定是否清除应用程序多维数据集中的现有数据。如果不想清除现有数据，则设置为 false。

表 3-27 （续） input.properties 参数

参数	说明
dataLoadValue	指定如何处理现有数据。如果想保留多维数据集中的现有数据，则指定 ADD_TO_EXISTING。

运行脚本

1. 通过复制上一节的脚本来创建 importData.ps1 或 importData.sh。
2. 创建 input.properties 文件，并将其保存在 importData 脚本所在的目录中。此文件的内容因操作系统的不同而异。请参阅“创建 input.properties 文件”。
   请确保您对此目录具有写权限。对于 Windows，您可能需要使用**以管理员身份运行**选项启动 PowerShell，以便能够运行脚本。
3. 启动脚本。
   - **Windows PowerShell**：运行 importData.ps1。
   - **Linux/UNIX**：运行 ./importData.sh。

Oracle Enterprise Data Management Cloud 示例方案

这些示例方案介绍了使用 EPM Automate 命令在 Oracle Fusion Cloud Enterprise Data Management 和 Oracle Fusion Cloud Enterprise Performance Management 之间同步应用程序维。

另请参阅：

- 将 Oracle Enterprise Data Management Cloud 维和映射与云 EPM 应用程序同步
  此示例方案介绍了在 Oracle Fusion Cloud Enterprise Data Management 应用程序和 Oracle Fusion Cloud Enterprise Performance Management 应用程序之间同步维。
- 将云 EPM 维与 Oracle Enterprise Data Management Cloud 应用程序同步
  此示例方案介绍了在 Oracle Fusion Cloud Enterprise Performance Management 应用程序和 Oracle Fusion Cloud Enterprise Data Management 应用程序之间同步维。

将 Oracle Enterprise Data Management Cloud 维和映射与云 EPM 应用程序同步

此示例方案介绍了在 Oracle Fusion Cloud Enterprise Data Management 应用程序和 Oracle Fusion Cloud Enterprise Performance Management 应用程序之间同步维。

使用本节中的脚本可完成以下任务：

- 从 Oracle Enterprise Data Management Cloud 应用程序中导出维
- 从 Oracle Enterprise Data Management Cloud 应用程序维中导出映射
- 将导出文件复制到云 EPM 环境
- 将维元数据和映射导入到云 EPM 应用程序

要在 Oracle Enterprise Data Management Cloud 应用程序与云 EPM 应用程序之间同步维和映射：

1. 通过复制以下脚本来创建脚本文件：

```
rem Integration example to sync application dimensions between Cloud EDM
and Cloud EPM
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 Cloud EPM 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. 修改脚本文件并设置所需的参数值。有关参数的说明和示例，请参阅“脚本执行的参数”。
3. 手动运行脚本或调度该脚本以根据需要运行。请参阅“自动执行脚本”。

脚本执行的参数

本节中的脚本文件要求您指定下表中说明的一些参数值。并非所有这些参数都会在所有脚本中使用。

表 3-28　脚本文件的参数值

参数	说明
EDMUSER	Oracle Enterprise Data Management Cloud 服务管理员的用户登录 ID。 **示例：** EDMUSER=jdoe@example.com
EDMSVR	Oracle Enterprise Data Management Cloud 环境的 URL。 **示例：** EDMSVR=https:// example.oraclecloud.com
EDMPWDFILE	Oracle Enterprise Data Management Cloud 服务管理员的加密密码文件 (EPW) 的名称和位置。 **示例：** EDMPWDFILE=edm_jdoe.epw
EDMAPP	Oracle Enterprise Data Management Cloud 应用程序维的名称。 **示例：** EDMAPP=USOperations
EDMDIM	要导出或导入的维的名称。 **示例：** EDMDIM=entity
EDMLOC	要导出的位置的名称。 **示例：** EDMLOC=Loc1
EPMUSER	云 EPM 服务管理员的登录名。 **示例：** EPMUSER=john.doe@example.com
EPMSVR	云 EPM 环境的 URL。 **示例：** EPMSVR=https://example.oraclecloud.com
EPMIMPJOB	云 EPM 环境中类型为导入元数据的现有导入作业的名称。 **示例：** EPMIMPJOB=imp_DIMMetadata
EPMEXPJOB	云 EPM 环境中类型为导出元数据的现有作业的名称。 **示例：** EPMEXPJOB=Exp_DIMMetadata
PWDFILE	云 EPM 服务管理员的加密密码文件 (EPW) 的名称和位置。请参阅 encrypt 命令。 **示例：** PWDFILE=pwd_jdoe.epw
DIMFILE	保存已导出维数据的文件的名称。 **示例：** DIMFILE=entity_file.CSV
MAPFILE	保存已导出映射数据的文件的名称。 **示例：** MAPFILE=map_file.CSV

将云 EPM 维与 Oracle Enterprise Data Management Cloud 应用程序同步

此示例方案介绍了在 Oracle Fusion Cloud Enterprise Performance Management 应用程序和 Oracle Fusion Cloud Enterprise Data Management 应用程序之间同步维。

使用本节中的脚本可完成以下任务：

- 从云 EPM 应用程序中导出元数据（维）
- 将包含维数据的导出文件复制到 Oracle Enterprise Data Management Cloud 环境
- 将维元数据导入到 Oracle Enterprise Data Management Cloud 应用程序

要在云 EPM 应用程序与 Oracle Enterprise Data Management Cloud 应用程序之间同步维：

1. 通过复制以下脚本来创建 Windows 脚本文件：

```
rem Integration example to sync an application dimension between Cloud EPM
and Cloud 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
```

   修改脚本文件并设置所需的参数值。有关参数的说明和示例，请参阅“脚本执行的参数”。

2. 手动运行脚本或调度该脚本以根据需要运行。请参阅“自动执行脚本”。

自动执行脚本

服务管理员在 Windows 任务计划程序中调度脚本执行时间，或者使用 EPM Automate 通过 cron 作业自动执行活动。

要使用 Windows 任务计划程序调度 EPM Automate 脚本执行：

1. 依次单击**开始**、**控制面板**和**管理工具**。
2. 打开**任务计划程序**。
3. 依次选择**操作**和**创建基本任务**。
4. 输入任务名称和可选说明，然后单击**下一步**。
5. 在**任务触发器**中，选择用于运行脚本的计划，然后单击**下一步**。
6. 在下一个屏幕中，指定其他调度参数，然后单击**下一步**。

7. 在“操作”中，确保选择**启动程序**。
8. 在**启动程序**中，完成以下步骤：
   a. 在**程序或脚本**中，浏览并选择要调度的脚本。
   b. 在**添加参数 (可选)** 中，输入由 `SET user` 脚本参数标识的服务管理员的密码。
   c. 在**起始于 (可选)** 中，输入安装 EPM Automate 的位置；通常为 `C:/Oracle/EPMAutomate/bin`。
   d. 单击**下一步**。
9. 在**摘要**中，选择**当单击“完成”时，打开此任务属性的对话框**，然后单击**完成**。
10. 在**常规**中，选择以下安全选项，然后单击**确定**。
    - **不管用户是否登录都要运行**
    - **使用最高权限运行**

监视 EPM Automate 活动

为了帮助您识别初始化的操作的状态，EPM Automate 将在您从中运行 EPM Automate 的控制台中显示状态代码。

请参阅“退出代码”。

使用作业控制台监视您使用 EPM Automate 执行的作业。有关详细信息，请参阅《管理 *Planning*》 指南中的“管理作业”。

4

在不安装 EPM Automate 的情况下运行命令

使用 Groovy，可以直接在 Oracle Fusion Cloud Enterprise Performance Management 中运行部分命令。无需安装 EPM Automate 即可运行服务器端命令。

> **Note:**
>
> 在此方案中，Groovy 脚本编写为可直接在云 EPM 中执行，而不是在任何客户端计算机上执行。

在本章中：

- 支持服务器端命令执行的环境
- 信息源
- 支持的命令
- 使用服务器端 Groovy 运行 EPM Automate 的方法
- 使用服务器端 Groovy 脚本克隆环境
- 使用服务器端 Groovy 脚本通过电子邮件发送活动报表

支持服务器端命令执行的环境

仅以下环境中支持编写在服务器端执行 EPM Automate 命令的 Groovy 脚本：

- 在 EPM Enterprise Cloud Service 环境中部署的 Planning 和 Planning 模块业务流程。
- Enterprise Planning and Budgeting Cloud
- 具有 Plus One 选件的 Planning and Budgeting Cloud
- 自由形式
- Enterprise Profitability and Cost Management
- EPM Enterprise Cloud Service 环境中的 Financial Consolidation and Close。
- EPM Enterprise Cloud Service 环境中的 Tax Reporting。
- 战略性人员规划
- 销售规划

只能在上述 Oracle Fusion Cloud Enterprise Performance Management 环境中编写和执行包含 EPM Automate 命令的 Groovy 脚本。但是，在此类环境中编写的脚本可以在任何云 EPM 环境中发出 EPM Automate 命令。例如，您可以在 Planning EPM Enterprise Cloud Service 环境中创建脚本，并且使用该脚本在不支持 Groovy 脚本编写的 Narrative Reporting 环境中发出命令。

信息源

有关详细信息，请参阅使用 *Calculation Manager* 进行设计中的以下源：

- 关于 Groovy 业务规则
- 创建 Groovy 业务规则

支持的命令

除以下命令外，所有 EPM Automate 命令都可以通过 Groovy 运行：

- downloadFile
- upgrade
- uploadFile

无法在运行 Groovy 脚本的环境中执行以下命令：

- recreate
- replay
- resetService
- restructureCube

> **Note:**
>
> - 运行 encrypt 命令时，将在服务器上创建加密的密码文件，并保留以供长期使用。如果运行 listFiles 命令，则不会列出加密的密码文件。无法使用 downloadFile 命令从服务器下载该文件。
> - 为了使 feedback 命令正常工作，用作附件的所有文件和屏幕截图都应在默认上传位置提供。请参阅“默认上传位置”。如果未在 uploadFile 命令中指定位置，则此位置为存储文件的位置。

使用服务器端 Groovy 运行 EPM Automate 的方法

- `getEPMAutomate ()` - 此静态方法提供 `EpmAutomate` 类的实例，然后可用于调用 EPM Automate 命令。
- `execute ()` - `EpmAutomate` 类的此方法用于执行 EPM Automate 命令。将 EPM Automate 命令名称作为第一个参数传递，并将命令选项作为后续参数传递。此方法返回 `EpmAutomateStatus` 类的实例。
- `getStatus ()` - `EPMAutomateStatus` 类的此方法返回命令返回的执行状态。返回值 0 表示成功，而非零值表示命令失败。
- `getOutput ()` - `EPMAutomateStatus` 类的此方法返回命令的字符串输出。例如，可以使用此方法返回 `getApplicationAdminMode` 和 `getDailyMaintenanceStartTime` 命令的输出。如果命令的返回状态为非零，则此方法返回错误消息。
- `getItemsList ()` - `EPMAutomateStatus` 类的此方法返回命令的列表输出。例如，可以使用此方法返回 `getSubstVar`、`listBackups` 和 `listFiles` 命令的输出。

使用服务器端 Groovy 脚本克隆环境

您可以在服务器端 Groovy 脚本中包含 EPM Automate 命令来克隆环境，从而实现灾难恢复。这允许在不占用内部部署空间的情况下设置灾难恢复。

如果密码中包含特殊字符，请参阅“处理特殊字符”。此外，还要确保替换这些参数值以适应您的环境：

Table 4-1　要更改的参数

参数	说明
password	在源环境中执行克隆操作的服务管理员的密码。
targetpassword	在目标环境中执行克隆操作的服务管理员的密码。
username	源环境中的服务管理员的用户 ID。
targetusername	目标环境中的服务管理员的用户 ID。此用户还必须分配有目标环境中的身份域管理员角色。
email_id	计划将有关克隆过程的信息发送到的电子邮件地址。

用于加密密码的脚本

```
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())
```

用于克隆环境的脚本

此脚本使用前面的脚本创建的加密密码文件。

```
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())
```

使用服务器端 Groovy 脚本通过电子邮件发送活动报表

此脚本可用于通过电子邮件将活动报表发送给收件人列表。然后可以调度此脚本每天运行，以获取每日活动报表。此脚本执行以下功能：

- 对执行 Groovy 脚本的服务管理员的密码进行加密。
- 使用加密密码登录 Oracle Fusion Cloud Enterprise Performance Management 环境。
- 通过电子邮件将环境中可用的活动报表发送给收件人列表，通常是服务管理员
- 从环境中注销。

如果密码中包含特殊字符，请参阅“处理特殊字符”。此外，还要确保替换这些参数值以适应您的环境：

Table 4-2　要更改的参数

参数	说明
user	要登录环境的服务管理员的用户 ID。
password	服务管理员的密码。
url	云 EPM 环境的 URL，将从该环境通过电子邮件发送活动报表。
emailaddresses	活动报表要发送到的电子邮件地址的分号分隔列表。

有关使用 Groovy 规则的详细信息，请参阅《管理 *Planning*》中的“使用 Groovy 规则”。

```
/*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

复制云 EPM 环境

配置辅助的 Oracle Fusion Cloud Enterprise Performance Management 环境时需要执行以下步骤，以便在主 Oracle 数据中心因不可预测的情况而变得不可用时确保服务的可用性。

> **注：**
>
> 本附录中论述的过程不适用于 Narrative Reporting。

- 设置每日对象复制
- 设置按需复制
- 配置辅助环境

设置每日复制

要复制环境，可以使用 EPM Automate 将在日常维护期间创建的对象快照从主环境复制到辅助环境。

Oracle 每天在每个环境中执行例行维护。在此服务维护期间，Oracle 通过备份环境的内容（现有数据和对象，包括来自身份域的用户和角色分配）来创建维护快照。

要设置每日服务复制：

1. 创建包含以下 EPM Automate 命令的脚本文件。此脚本将应用程序快照从主环境复制到辅助环境。

   > **注：**
   >
   > 确保更改用户名、密码文件、身份域名和服务 URL。有关创建加密密码文件的信息，请参阅 encrypt 命令。

   ```
   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. 使用调度程序（Windows 任务计划程序）调度脚本文件的执行时间，使其从维护期间开始时运行两个小时。

3. 在主环境和辅助环境中设置相同的维护期间开始时间。有关详细信息，请参阅《管理员入门指南》中的“设置服务维护时间”。

设置按需复制

为减少 RPO，可以创建主环境的按需快照，然后将它们复制到辅助环境。

例如，可以创建一个 EPM Automate 脚本，安排该脚本在例行的每日复制之间每六个小时运行一次，从而将 RPO 从 24 小时减少为六小时。

> **注：**
>
> 在创建按需快照过程中，主环境将被置于只读模式几分钟。

要设置按需复制：

1. 创建包含以下 EPM Automate 命令的脚本文件。此脚本将应用程序快照从主环境复制到辅助环境。

   > **注：**
   >
   > 确保更改用户名、密码文件、身份域名和服务 URL。有关创建加密密码文件的信息，请参阅 encrypt 命令。

```
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. 使用调度程序（例如 Windows 任务计划程序）调度脚本文件的执行时间，使其按需运行以满足所需的 RPO。

配置辅助环境

配置辅助环境以将其激活。

仅当主环境在相当长的一段时间内不可用而需要激活辅助环境时，才需要完成此过程。在配置辅助环境之前，请参阅《管理员入门指南》 中的以下主题：

- 旧 EPM 云快照的迁移路径

- EPM Standard Cloud Service 和 EPM Enterprise Cloud Service 快照的迁移路径

要配置辅助环境：

1. 启动 EPM Automate 会话并完成以下活动。
   - 使用一个同时具有服务管理员和身份域管理员角色的帐户登录到辅助环境。确保指定适当的用户名、密码、域名和服务 URL。
   - 重新创建辅助环境。
     - 如果主环境是 Planning、Tax Reporting、Financial Consolidation and Close 或 Enterprise Profitability and Cost Management 环境，请使用：
       `epmautomate recreate -f`
     - 如果主环境不是 Planning、Tax Reporting、Financial Consolidation and Close 或 Enterprise Profitability and Cost Management 环境，请使用：
       `epmautomate recreate -f TempServiceType=PRIMARY_APPLICATION_TYPE`, where `PRIMARY_APPLICATION_TYPE` is `ARCS`, `EDMCS`, `PCMCS`, or `EPRCS`。
   - 从快照导入应用程序和身份域对象。
   - 注销。

   可以通过运行以下命令来完成前述活动。请参阅以下主题：
   - login 命令
   - recreate 命令
   - importSnapshot 命令

   ```
   epmautomate login serviceAdmin secondaryPassword.epw secondary_URL
   epmautomate recreate –f
   epmautomate importsnapshot "Artifact Snapshot" importUsers=true
   epmautomate logout
   ```

2. 登录到辅助环境并验证是否所有数据都可用。
3. 将辅助环境的 URL 发送给所有用户。

A

准备运行 simulateConcurrentUsage 命令

simulateConcurrentUsage 目录支持在环境中执行以下操作来模拟负载：

- 打开表单
- 保存表单
- 运行业务规则
- 运行业务规则集
- 运行数据规则
- 打开即席网格
- 执行管理报表工作簿
- 执行管理报表报表

涉及的步骤

1. 创建 `requirement.csv` 文件。请参阅“创建 requirement.csv 文件”
2. 创建输入文件，在其中指定 `requirement.csv` 中包含的操作的详细信息。请参阅：
   - 打开表单输入文件
   - 保存表单输入文件
   - 运行业务规则输入文件
   - 运行业务规则集输入文件
   - 运行数据规则输入文件
   - 即席网格输入文件
   - 执行工作簿输入文件
   - 执行报表输入文件
3. 如果需要设置用户变量成员映射，则创建包含用户变量详细信息的 `UserVarMemberMapping.csv`。请参阅“创建 UserVarMemberMapping.csv 文件”
4. （可选）如果需要使用 Smart View 选项，则将 Oracle Smart View for Office 选项导出到 `options.xml` 中。有关详细信息，请参阅“创建 options.xml 文件”。
5. （可选）创建 `users.csv` 文件以使用现有用户运行模拟。有关详细信息，请参阅“创建 users.csv 文件”。
6. 创建包含在前面步骤中创建的文件的 ZIP 文件，并将其上传到环境。请参阅“创建输入 ZIP 文件并将其上传到环境”
7. 使用上传的 ZIP 文件运行 simulateConcurrentUsage 命令。

创建 requirement.csv 文件

首先创建 `requirement.csv` 文件，在其中列出要测试的用例的详细信息。此 CSV 文件中的每一行标识要执行的操作类型、对象名称、并发用户数、指定操作详细信息的输入文件以及与操作相

关的其他信息（如果有）。例如，要打开 2 个表单、保存 2 个表单以及运行 2 个业务规则，则在输入 CSV 文件中指定 6 行。`requirement.csv` 的第一行必须包含以下信息：

```
#Type of Operation,Artifact Name,Number of Users,Input File,Additional Info
```

该文件的后续每一行都包含单个操作及其参数。一些操作可能不需要所有这些参数值。下表中解释了要求的文件条目。

> **Note:**
>
> 除非在表中另外指明，否则需要所有值。

Table A-1 requirement.csv 格式

字段	说明
Type of operation（操作类型）	必须为以下项之一： • `Open Form`（打开表单） • `Save Form`（保存表单） • `Run Business Rule`（运行业务规则） • `Run Business Ruleset`（运行业务规则集） • `Run Data Rule`（运行数据规则） • `Ad Hoc Grid`（即席网格） • `Execute Report`（执行报表） • `Execute Book`（执行工作簿）
Artifact Name（对象名称）	此值取决于操作类型： • Open Form（打开表单）：要打开的表单的名称和位置。 • Save Form（保存表单）：要保存的表单的名称和位置。 • Run Business Rule（运行业务规则）：业务规则的名称。 • Run Business Ruleset（运行业务规则集）：业务规则集的名称。 • Data Rule（数据规则）：数据规则的名称。 • Ad Hoc Grid（即席网格）：不适用（留空）。 • Execute Report（执行报表）：报表的名称和位置。 • Execute Book（执行工作簿）：工作簿的名称和位置。
Number of Users（用户数）	要模拟并发使用的用户数。
Input File（输入文件）	包含以下内容的 CSV 文件的名称：POV 值、运行时提示或要使用的特定于用例的其他值
Additional Info（其他信息）	操作所需的其他参数。仅适用于即席网格。对于其他用例，则将此项留空。

注：对象名称必须与应用程序中的对象名称匹配，并且大小写必须相同。

`requirement.csv` 文件的示例：

```
# 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 Business Ruleset,RollupUSSales,5,runbusinessruleset_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,
```

创建输入文件

requirement.csv 中标识的每个用例都必须有一个匹配的输入文件，以提供执行用例所需的所有参数。

理想情况下，输入文件包含的条目数必须与在 requirement.csv 中为此用例指定的用户数一致。

如果输入文件中的条目数少于 requirement.csv 中该用例的并发用户数，则 EPM Automate 将重复使用输入文件中的一些条目来运行用例，直到按 requirement.csv 中标识的用户数执行了操作为止。

例如，如果 requirement.csv 中“运行业务规则”操作的用例条目如下：

```
Run Business Rule, Copy Budget,10,br_input_file.csv,
```

br_input_file.csv 还应包含 10 个条目。如果 br_input_file.csv 仅包含六个条目，则 EPM Automate 将其用于前 6 个用户。对于后 4 个用户，EPM Automate 将重复使用 br_input_file.csv 中的前 4 个条目。

如果输入文件中的条目数超过为用例指定的用户数，则 EPM Automate 忽略输入文件中最后多余的条目。

- 打开表单输入文件
- 保存表单输入文件
- 运行业务规则输入文件
- 运行业务规则集输入文件
- 运行数据规则输入文件
- 即席网格输入文件
- 执行报表输入文件
- 执行工作簿输入文件

打开表单输入文件

此文件（在 requirement.csv 中引用以支持打开表单）包含以下格式的 POV 条目：pov=[DIM 1:MEMBER 1],[DIM 2:MEMBER 2],[DIM 3:MEMBER 3], 等。

在本次讨论中，DIM 1、DIM 2 等是维名称，MEMBER 1、MEMBER 2 等是它们在 POV 中的维成员值。

示例输入文件：

```
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:**
>
> 如果您在 requirement.csv 中指定的表单要求设置用户变量，则还必须创建 UserVarMemberMapping.csv。请参阅“创建 UserVarMemberMapping.csv 文件”。

保存表单输入文件

此文件（在 requirement.csv 中引用以支持保存表单）包含 POV 和单元格输入值，格式如下。如果业务规则与表单关联，则此文件还包含这些业务规则的运行时提示：

示例输入文件：

```
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]]..
```

在此示例中：

- DIM 表示维名称，MEMBER 表示维成员值。
- CELL COLUMN HEADER 标识列标题的名称，CELL ROW HEADER 标识行标题的名称。
- BUSINESS RULE NAME 指示业务规则的名称，RTP 指示运行时提示名称，VALUE 指示其值。如果没有业务规则与表单关联或者您想要使用表单中存在的默认运行时提示值，则不需要 RTP。

例如：

```
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:**
>
> 如果您在 requirement.csv 中指定的表单要求设置用户变量，则还必须创建 UserVarMemberMapping.csv。请参阅“创建 UserVarMemberMapping.csv 文件”。

与表单关联的智能推送（如果有）会自动运行。

运行业务规则输入文件

此文件（在 requirement.csv 中引用以支持运行业务规则）包含以下格式的运行时参数值：rtp=*[PARAMETER:VALUE],[PARAMETER:VALUE]* 等。在此格式中，*PARAMETER* 标识运行时提示名称，*VALUE* 标识其值。

务必根据需要添加尽可能多的运行时提示以运行规则。如果没有提供运行时提示及其值，则使用默认值。该命令忽略与为规则定义的提示不完全匹配的运行时提示。如果业务规则不需要运行时参数，则包含 rtp=[]。

示例输入文件：

```
rtp=[Period:Q1],[Entity:USA]
rtp=[Period:Q2],[Entity:USA]
rtp=[Period:Q3],[Entity:USA]
rtp=[Period:Q4],[Entity:USA]
```

运行业务规则集输入文件

此文件（在 requirement.csv 中引用以支持运行业务规则集）包含以下格式的运行时参数值：rtp=[*PARAMETER:VALUE*],[*PARAMETER:VALUE*] 等。在此格式中，*PARAMETER:VALUE* 标识运行时参数名称及其值。

使用 *PARAMETER:VALUE* 对可以指定规则集所需的任意数量的运行时提示。如果您没有为运行时参数提供值，则使用默认值。该命令忽略与为规则集定义的提示不完全匹配的运行时提示。如果业务规则集不需要运行时参数值，则包含 rtp=[]。

示例输入文件：

```
rpt=[Period:Q1],[Entity:USA]
rtp=[Period:Q2],[Entity:USA]
rtp=[Period:Q3],[Entity:USA]
rtp=[Period:Q4],[Entity:USA]
```

运行数据规则输入文件

此文件（在 requirement.csv 中引用以支持运行数据规则）应指定起始期间、结束期间、导入模式、导出模式和环境中可用的可选导入文件名。如果未指定文件名，则将使用数据规则中指定的文件名。每一行的格式：startperiod=*START PERIOD*;endperiod=*END PERIOD*;importmode=IMPORT_MODE;exportmode=*EXPORT_MODE*;filename=*FILE NAME*

示例输入文件：

```
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
```

即席网格输入文件

simulateConcurrentUsage 命令支持本地模式和标准模式即席网格。在本地模式网格中，POV 显示为 Oracle Smart View for Office 插件工具栏的一部分。在标准模式下，POV 是电子表格本身的一部分，并占据电子表格的第一行。

即席网格输入文件（在 `requirement.csv` 中引用以支持打开即席网格）应指定要打开的网格。文件中的每一行都应采用以下格式。

```
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,..]
```

示例输入文件：

```
fileName=dropdown.xlsx#sheet4;pov=[HSP_View:BaseData],[Scenario:Forecast],
[Product:No Product],[Entity:Sales Mid-Atlantic];rows = [ Account ]; cols=
[Year, Period, Version]
```

> **Note:**
>
> 输入 CSV 文件中 `fileName` 标识的文件必须包含指定工作表中的即席网格定义。例如，上面的示例输入文件指定 `dropdown.xlsx` 的 `sheet4` 作为即席网格定义的源。用于运行 simulateConcurrentUsage 命令的 *INPUT_FILE.zip* 中必须有此 Excel 文件以及 `requirement.csv` 和输入 CSV 文件。

执行报表输入文件

此文件（在 `requirement.csv` 中引用以支持打开管理报表）应指定要打开的报表。文件中的每一行都应采用以下格式。

```
format=[REPORT_FORMAT];globalPov=[DIM 1:MEMBER 1],[DIM 2:MEMBER
2],..;prompts=[PROMPT 1:VALUE 1],[PROMPT 2:VALUE 2],..
```

`globalPov` 和 `prompts` 是可选项。

> **Note:**
>
> - 支持的报表格式为 `pdf` 和 `embedded`。
> - 如果 globalPov 维的名称或其成员包含冒号 (:) 或分号 (;)，则需要在其前面使用转义字符 \\。例如，维名称 `Version:View` 应指定为 `Version\\:View`

用于使用全局 POV `[Version View:Working],[Sales Entity:International Sales]` 和提示 `[Actual;Budget],[Year:2018]` 基于报表生成 pdf 的示例输入文件：

```
format=pdf;globalPov=[Version View:Working],[Sales Entity:International
Sales];prompts=[Actual:Budget],[Year:2018]
```

执行工作簿输入文件

此文件（在 `requirement.csv` 中引用以支持在“报表”中打开工作簿）应指定要打开的工作簿。文件中的每一行都应采用以下格式。

`format=BOOK_FORMAT` 或

`format=BOOK_FORMAT;globalPov=[DIM 1:MEMBER 1],[DIM 2:MEMBER 2],..`

用于使用全局 POV *[Version View:Working],[Sales Entity:International Sales]* 基于工作簿生成 pdf 的示例输入文件：

```
format=pdf;globalPov=[Version View:Working],[Sales Entity:International Sales]
```

> **Note:**
>
> - 支持的工作簿格式为 PDF 和 XLSX。
> - 如果 globalPov 维的名称或其成员包含冒号 (:) 或分号 (;)，则需要在其前面使用转义字符 \\。例如，维名称 `Version:View` 应指定为 `Version\\:View`

创建 UserVarMemberMapping.csv 文件

如果在“打开表单”或“保存表单”用例的输入文件中指定的表单要求设置用户变量，则需要此文件。其他用例不需要此文件。

此文件的第一行是标题 `#Dimension,User Variable,Member`

后续条目包含维的映射、用户变量和维成员。

示例 `UserVarMemberMapping.csv` 文件：

```
#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
```

创建 options.xml 文件

模拟打开表单、保存表单和即席网格用例时，可以使用 Oracle Smart View for Office 选项，例如隐藏行、隐藏缺少的块和页成员缩进。这些选项中的每个选项都会导致在环境中产生不同程度的负载。

如果要使用 Smart View 选项模拟负载，可以在输入 ZIP 文件中包括 `options.xml` 文件。

可以通过导出 Smart View 选项创建 `options.xml`。要生成 `options.xml`，请从 **SmartView** 选项卡中，依次单击**选项**、**确定**和**导出选项**。

options.xml 的使用不是必需的。如果输入 ZIP 文件中不存在此文件，则将使用默认选项进行模拟。

创建 users.csv 文件

simulateConcurrentUsage 命令的模式 4 允许您使用输入 ZIP 文件中包含的 users.csv 文件中标识的现有用户运行该命令。此模式不会为模拟创建用户。

users.csv 文件的格式为 loginname,password：

```
jdoe,jdoe_pwd
john.doe@example.com,john_doe_pwd
```

users.csv 中 loginname 的值必须与身份域中定义的登录名匹配。此外，标识的用户必须具有运行该操作所需的预定义角色和应用程序角色。

创建输入 ZIP 文件并将其上传到环境

如果需要，使用 7 Zip 等工具，创建一个包含以下内容的 ZIP 文件：requirement.csv、对应的用例输入文件和（可选）UserVarMemberMapping.csv、users.csv 和 options.xml。

使用 uploadFile 命令将生成的 ZIP 文件上传到要运行模拟的环境的收件箱中（示例命令语法 epmautomate uploadFile "C:/uploads/*INPUT_FILE.zip*" inbox）。

"Simulate Concurrent Usage Report"（模拟并发使用报表）示例

默认情况下，将向执行 simulateConcurrentUsage 命令的用户发送 "Simulate Concurrent Usage Report"（模拟并发使用报表）。如果您指定电子邮件收件人，则将仅向这些电子邮件收件人发送该报表。

Simulate Concurrent Usage report

Operation #	Operation	Artifact Name	Users	Iterations	Min. Duration	Max. Duration	Avg. Duration	Return Status
1	Open Form	Set Services Revenue Forecast Assumptions	2	1	00:00:00.17	00:00:00.21	00:00:00.19	Passed
2	Save Form	Set Headcount and Salary Forecast Assumptions	1	1	00:00:00.45	00:00:00.45	00:00:00.45	Passed
3	Run Business Rule	Operating Expense Adj Plan	2	1	00:00:00.60	00:00:00.61	00:00:00.60	Passed
4	Run Data Rule	test	2	1	00:00:03.00	00:00:03.29	00:00:03.14	Passed
5	Ad Hoc Grid		1	1	00:00:00.13	00:00:00.13	00:00:00.13	Passed
6	Execute Book	Book1	1	1	00:00:10.21	00:00:10.21	00:00:10.21	Passed
7	Execute Report	Variance Explanations1	1	1	00:00:04.95	00:00:04.95	00:00:04.95	Passed
8	Save Form	Set Services Revenue Forecast Assumptions	1	1	00:00:05.56	00:00:05.56	00:00:05.56	Passed
9	Run Business Rule	Allocate Plan Targets	2	1	00:00:02.62	00:00:02.62	00:00:02.62	Passed
10	Run Business Ruleset	Revenue Plan	2	1	00:00:00.60	00:00:00.61	00:00:00.61	Passed

此报表标识以下内容：

列	说明
Operation #（操作编号）	requirement.csv 中的用例的序号

列	说明
Operation（操作）	`requirement.csv` 中指定的操作类型
Artifact Name（对象名称）	`requirement.csv` 中指定的对象名称
Users（用户）	`requirement.csv` 中指定的用户数
Iterations（迭代次数）	`iterations` 参数指定的用例执行次数
Min. Duration（最短持续时间）	一个用户执行此用例所需的最短时间
Max. Duration（最长持续时间）	一个用户执行此用例所需的最长时间
Avg. Duration（平均持续时间）	一个用户执行此用例所需的平均时间
Return Status（返回状态）	用例的状态。如果用例执行未成功，则显示 `Failed`（失败）

B

准备运行 Replay 命令

replay 命令用于对在一定负载下的环境执行性能测试，以验证当服务处于指定负载下时用户体验是否能达到可接受的水平。需要先完成几个步骤才能加载测试环境。

本附录介绍服务管理员在运行 EPM Automate 命令 replay 之前必须完成的步骤。

- 关于 Replay 命令
- 先决条件
- 创建 HAR 文件
- 创建重放文件
- 生成跟踪文件
- 重放示例会话

关于 Replay 命令

Replay 命令可在环境中重放 Oracle Smart View for Office、Oracle Fusion Cloud Enterprise Performance Management REST API 或 EPM Automate 负载以在重负载下进行性能测试，从而验证当环境处于指定负载下时用户体验是否可接受。

例如，可以测试重负载下测试环境的用户体验，从而确保在将应用程序从测试环境迁移到生产环境后环境运行良好。

先决条件

使用重放文件执行命令时，EPM Automate 并行运行重放文件中的每一行以在服务上产生负载，使您可以执行测试来验证服务在此负载下时用户体验是否可接受。

- 找出需要在环境中执行大量处理操作的表单。处理大量数据的表单或包括复杂计算的表单是很好的选择。例如，用于提交预测的表单、涉及创建即席和静态报表进程的表单可能会在服务上产生重负载。类似地，诸如运行业务规则、运行报表，以及执行资源密集型 REST API 和 EPM Automate 命令（例如 runBusiness rule、runDataRule、exportData、exportMetadata、restructureCube）等活动可能会使环境承受重负载，因此可以作为负载测试的选择。
- 如果需要，安装 Fiddler。EPM Automate 需要 HTTP 存档格式 (HTTP Archive format, HAR) 1.1 文件，该文件包含 Oracle Smart View for Office、Oracle Fusion Cloud Enterprise Performance Management REST API 或 EPM Automate 与云 EPM 环境的交互记录。通常，使用 Fiddler 生成 HAR 文件，以捕获与云 EPM 的交互日志。
- 运行您先前找出的高负载活动。使用 Smart View 运行活动（例如打开和保存表单、运行业务规则以及创建报表），然后使用 Fiddler 捕获活动详细信息并将其导出到 HAR 文件。类似地，运行 REST API 和 EPM Automate 命令并使用 Fiddler 捕获详细信息。有关详细信息，请参阅“创建 HAR 文件”。
- 创建的重放 CSV 文件列出了凭据（用户名和密码）和要运行的 HAR 文件的名称。该文件中的每一行包含一个不重复用户的用户名和密码，以便模拟同时进行多个用户会话。有关详细信息，请参阅“创建重放文件”。

在某行中指定了凭据来运行 HAR 文件的用户不必是运行会话创建 HAR 文件的用户。但是，此用户应有权在环境中运行这些活动。

有关 replay 命令的详细信息，请参阅“重放示例会话”。

创建 HAR 文件

HAR 文件捕获 Oracle Smart View for Office、REST API 或 EPM Automate 与 Oracle Fusion Cloud Enterprise Performance Management 的交互跟踪。

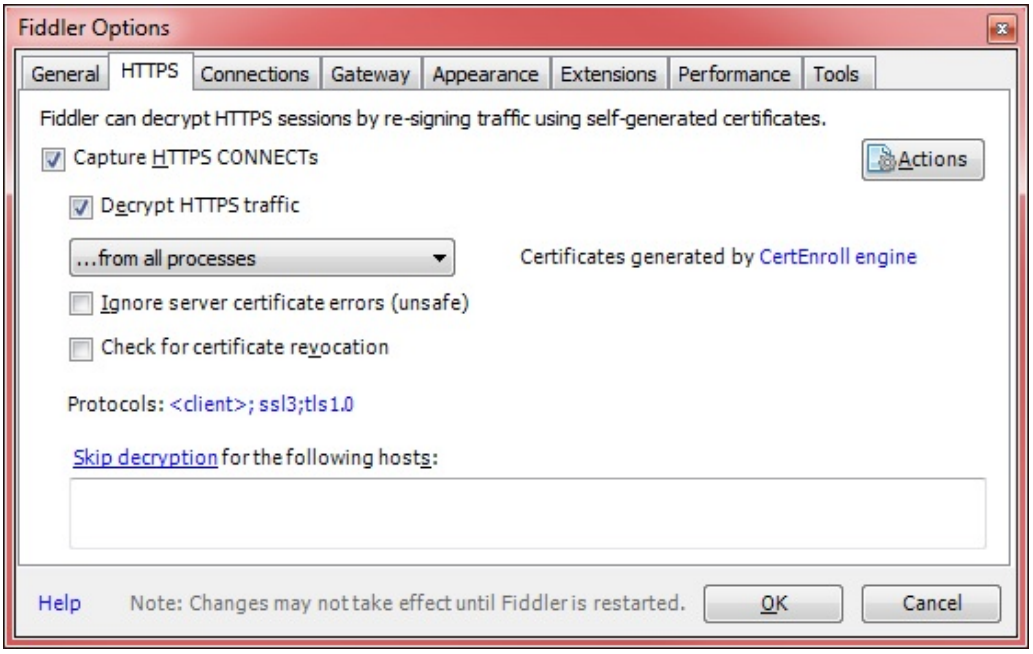
因为 Fiddler 捕获所有 HTTP(S) 通信的信息，所以创建 HAR 文件时，应禁止可能向 Fiddler 添加不必要的跟踪信息的活动。

要创建 HAR 文件：

1. 启动 Fiddler。
2. 确保 Fiddler 配置为解密所有进程中的 HTTPS 通信。
   a. 依次选择**工具**、**选项**和 **HTTPS**。
   b. 选择**解密 HTTPS 通信**，如果未选择的话。

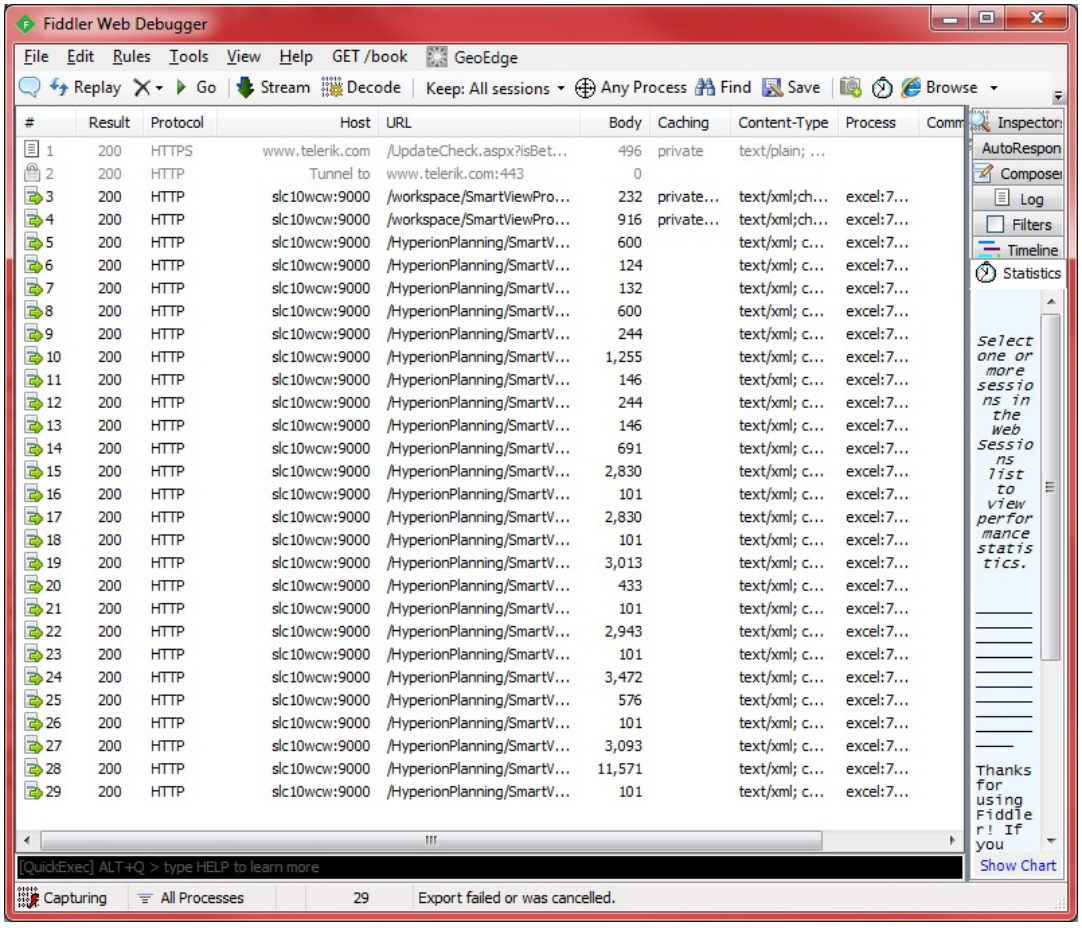
      Fiddler 显示用于拦截 HTTPS 通信的根证书的信息。信任此证书通常不会有问题。

   c. 如果要将根证书添加到可信 CA 列表，请单击**是**；否则选择**否**。
   d. **可选**：如果在上一步中选择了**否**，可以选择**忽略服务器证书错误**来隐藏与解密 HTTPS 通信相关的 Fiddler 安全警告。
   e. 单击**确定**。



3. 启动 Smart View 并访问要收集其跟踪信息的环境。
4. 使用 Smart View、REST API 或 EPM Automate，执行在环境中产生高处理负载的活动。例如，打开 Smart View 中的表单，以便 Fiddler 可以记录您的活动。

   Fiddler 记录您启动的进程。

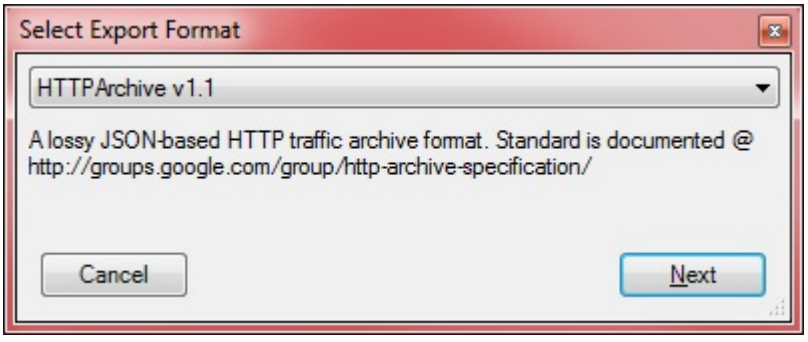


5. 在 Fiddler 中，完成以下步骤：

   a. 依次选择文件和导出会话，然后选择所有会话或选定会话。如果运行 Fiddler 时您已连接到其他网站，则选择选定会话来选择与环境相关的会话。

   b. 在选择导出格式中，选择 `HTTPArchive v1.1` 作为导出格式。

   c. 单击下一步。



   d. 在导出为 `HTTPArcive v1.1` 中，选择要存储文件的目录并指定文件名。

   e. 单击保存。

创建重放文件

重放文件是 CSV 文件，其中列出了凭据（用户名和密码）和要使用 EPM Automate 命令 replay 在系统上产生负载的 HAR 文件的名称。

确保指定的用户名和密码有权运行 HAR 文件中包括的活动。

执行 replay 命令时，EPM Automate 并行运行重放文件中的每一行以在服务上产生负载。例如，如果重放文件包含 10 行，则 EPM Automate 重放 10 个会话，从而您可以执行测试来验证服务在指定负载下时用户体验是否可接受。HAR 文件中包含的每个活动是连续运行的。

有关运行 replay 命令的信息，请参阅“replay ”。

要创建重放文件：

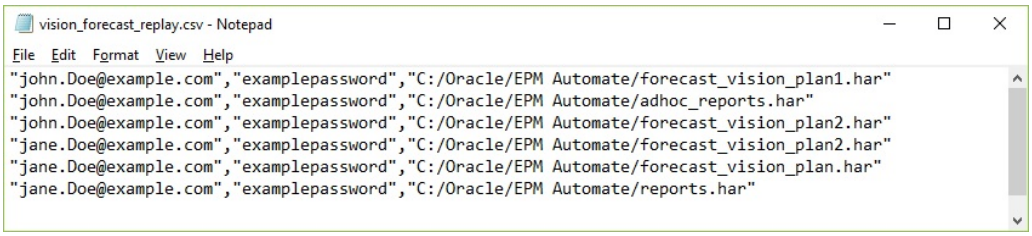
1. 打开 Microsoft Office Excel 并启动新的工作表。
2. 在第 1 行的 A、B 和 C 列中分别输入用户名、命令和 HAR 文件的位置。

   重复此步骤来创建其他行。

   > **注：**
   >
   > 必须指定 HAR 文件位置的绝对路径。请使用斜杠 (/) 作为文件路径中的目录分隔符；不要使用反斜杠 (\)。

3. 保存文件
4. 在另存为中，完成以下步骤：
   a. 选择要存储重放文件的目录。
   b. 在文件名中指定名称，在另存为类型中选择 `CSV （逗号分隔） (*.csv)`。
   c. 单击保存。

      重放示例文件如下：



生成跟踪文件

运行 replay 命令时，可以生成跟踪文件来与 Oracle 技术支持共享，以便解决问题。Oracle 技术支持使用跟踪文件来了解环境是如何处理 Oracle Smart View for Office 活动的。

您可以将 `trace=true` 可选参数与 replay 命令配合使用来生成 XML 格式的跟踪文件。如果为 HAR 文件中的每个活动使用此参数，EPM Automate 会创建跟踪文件，在其中包含 Smart View 对该活动的响应。

跟踪文件命名为 `trace-N.xml`；例如 `trace-1.xml`，其中 `N` 是从 1 开始的一个计数值。如果在重放文件中指定多个名称相同的 HRA 文件，EPM Automate 会在一个文件夹中合并跟踪文件。

与 HAR 文件相关的跟踪文件存储在您运行 EPM Automate 的目录内的一个文件夹中。EPM Automate 为重放文件中列出的每个 HAR 文件创建一个文件夹。EPM Automate 以 `YYYY_MM_DD_HH_MM_SS_HAR_FILE_NAME` 格式，组合使用当前服务器系统时间与 HAR 文件名来命名文件夹。例如，如果 HAR 文件名是 `forecast1.har`，则文件夹名称可能是 `2016_06_08_10_21_42_forecast1`。

重放示例会话

介绍如何使用多个 HAR 文件运行重放命令。

本节假设满足以下条件：

- 您创建了以下 HAR 文件。每个 HAR 文件可以包含一组相同活动。有关详细信息，请参阅“创建 HAR 文件”。
  - `C:\Oracle\EPM Automate\forecast_vision_plan1.har`
  - `C:\Oracle\EPM Automate\forecast_vision_plan2.har`
  - `C:\Oracle\EPM Automate\forecast_plan2.har`
- 您创建了一个重放文件 `C:/Oracle/EPM Automate/vision_forecast_replay.csv`，其中包含以下内容（有关详细信息，请参阅“创建重放文件”）：

> **注：**
>
> 在重放文件中，请使用斜杠 (/) 作为文件路径中的目录分隔符；不要使用反斜杠 (\)。

```
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
```

要运行 replay 命令：

1. 在命令提示符窗口中，导航到安装了 EPM Automate 的相应目录；例如 `C:\Oracle\EPM Automate\bin`。
2. 以服务管理员身份登录到环境，然后执行 replay 命令：

   ```
   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 在控制台中显示重放信息，并在指定持续时间（上一示例中为 12 分钟）后结束处理。它还创建跟踪文件夹和文件，因为前面的命令中包括 `trace=true` 参数。
   因为该命令是从 `C:\Oracle\EPM Automate\bin` 执行的，所以 EPM Automate 在以下文件夹中存储跟踪文件。请注意，这些文件夹是根据 HAR 文件名命名的。

- 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. 从环境中注销：

```
epmautomate logout
```

C

处理特殊字符

Oracle Fusion Cloud Enterprise Performance Management 密码、代理密码和命令参数值可能包含特殊字符。需要进行特殊处理，EPM Automate 才能处理此类字符。

本节中的示例使用示例密码来说明如何使用特殊字符。

Oracle 建议您使用双引号将参数和值对括起来。

Windows

必须使用双引号 (") 括住特殊字符或包含特殊字符的参数值对这些特殊字符进行转义。

> **注：**
>
> 不能从名称中包含 `&` 的文件夹运行 EPM Automate；例如 `C:\Oracle\A&B`。

表 C-1　特殊字符处理：Windows

字符	说明	转义示例
)	右括号	• `Example")"pwd1` 或 • `"Example)pwd1"`
<	小于	• `Example"<"pwd1` 或 • `"Example<pwd1"`
>	大于	• `Example">"pwd1` 或 • `"Example>pwd1"`
&	& 符号	• `Example"&"pwd1` 或 • `"Example&pwd1"`
|	竖线	• `Example"|"pwd1` 或 • `"Example|pwd1"`
"	引号	• `Example"""pwd1` 或 • `"Example"pwd1"`

在 Windows 批处理文件中的明文密码中使用感叹号

在用于 EPM Automate 的 Windows 批处理文件中的明文密码中使用感叹号 (!) 时，应按如下所示进行处理：

1. 在感叹号之前使用两个脱字号 (^^) 作为转义字符。例如，如果密码为 `Welc0me!`，则将其编码为 `Welc0me^^!`
2. 通过包括以下声明，更新批处理文件以在文件开头设置 DisableDelayedExpansion：`setlocal DisableDelayedExpansion`
3. 删除脚本中的 `setlocal EnableExtensions EnableDelayedExpansion` 声明（如果存在）。

UNIX/Linux

在 UNIX 和 Linux 操作系统上，必须使用反斜杠 (\) 对这些特殊字符进行转义。

> **注：**
>
> - 要对 ！（感叹号）进行转义，请用单引号将密码引起来或者使用反斜杠 (\) 作为转义符。
> - 要对 \、$、' 和 " 进行转义，请用双引号将密码括起来或者使用反斜杠 (\) 作为转义符。

表 C-2　特殊字符处理：UNIX/Linux

字符	说明	转义示例
(	左括号	`Example\(pwd1`
)	右括号	`Example\)pwd1`
<	小于	`Example\<pwd1`
>	大于	`Example\>pwd1`
`	撇号	`Example\'pwd1`
!	感叹号	• `'Example!pwd1'` 或 • `Example\!pwd1`
#	井号	`Example\#pwd1`
&	& 符号	`Example\&pwd1`
|	竖线	`Example|pwd1`
;	分号	`Example\;pwd1`
.	句点	`Example\.pwd1`
"	引号	• `Example\"pwd1` 或 • `"Example\"pwd1"`
'	单引号	• `Example\'pwd1` 或 • `"Example\'pwd1"`
$	美元符号	• `Example\$pwd1` 或 • `"Example\$pwd1"`
\	反斜杠	• `Example\\pwd1` 或 • `"Example\\pwd1"`

在 UNIX 或 Linux 脚本中的明文密码中使用感叹号

在 UNIX/Linux 脚本中，如果存储在 shell 变量中的 EPM Automate 密码包含特殊字符，请使用三个反斜杠作为转义序列，然后用双引号将字符串括起来。例如，包含在 shell 变量 `password` 中的密码 `lzi[ACO(e*7Qd)jE` 应当按如下方式编写脚本：

```
password="lzi[ACO\\\(e*7Qd\\\)jE"
```

D

特定于每个云 EPM 服务的命令

- Account Reconciliation 命令
- Financial Consolidation and Close 命令
- Narrative Reporting 命令
- Oracle Enterprise Data Management Cloud 命令
- Planning、Planning 模块、自由形式、战略性人员规划和销售规划命令
- Profitability and Cost Management 命令
- Enterprise Profitability and Cost Management 命令
- Tax Reporting 命令

Account Reconciliation 命令

适用于 **Account Reconciliation** 的 **EPM Automate** 命令		
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
copyFromSFTP	importPreMappedTransactions	runDailyMaintenance
copySnapshotFromInstance	importProfiles	runDataRule
copyToObjectStorage	importRates	runDMReport
copyToSFTP	importRCAttributeValues	runIntegration
createGroups	importReconciliationAttributes	runMatchingReport
createReconciliations	importSnapshot	sendMail
deleteFile	importTMAttributeValues	setApplicationAdminMode
deleteGroups	importTmPremappedTransactions	setDailyMaintenanceStartTime
downloadFile	invalidLoginReport	setDemoDates
encrypt	listBackups	setEncryptionKey
exportAccessControl	listFiles	setIdleSessionTimeout
exportARApplicationProperties	login	setIPAllowlist
exportBackgroundImage	logout	setRestrictedDataAccess
exportDataManagement	provisionReport	setManualDataAccess
exportLogoImage	purgeArchivedTmTransactions	setPeriodStatus
exportMapping	purgeTmTransactions	setVirusScanOnFileUploads
exportSnapshot	recreate	skipUpdate
feedback	refreshCube	unassignRole
getApplicationAdminMode	removeUserFromGroups	updateUsers
getDailyMaintenanceStartTime	removeUsers	upgrade
getIdleSessionTimeout	removeUsersFromGroup	uploadFile
getIPAllowlist	removeUsersFromTeam	userAuditReport
getRestrictedDataAccess		userGroupReport
getVirusScanOnFileUploads		

Financial Consolidation and Close 命令

适用于 Financial Consolidation and Close 的 EPM Automate 命令

addUsers
addUsersToGroup
addUsersToTeam
addUserToGroups
applicationAdminMode
assignRole
clearDataByProfile
cloneEnvironment
copyDataByProfile
copyFileFromInstance
copyFromObjectStorage
copyFromSFTP
copyOwnershipDataToNextYear
copySnapshotFromInstance
copyToObjectStorage
copyToSFTP
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
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
runPipeline
runSupplementalDataReport
runTaskManagerReport
sendMail
setApplicationAdminMode
setDailyMaintenanceStartTime
setDemoDates
setEJJournalStatus
setEncryptionKey
setEssbaseQryGovExecTime
setIdleSessionTimeout
setIPAllowlist
setRestrictedDataAccess
setVirusScanOnFileUploads
setManualDataAccess
setSubstVars
simulateConcurrentUsage
skipUpdate
snapshotCompareReport
unassignRole
updateGuidedLearningSettings
updateUsers
upgrade
uploadFile
userAuditReport
userGroupReport

适用于 **Financial Consolidation and Close 的 EPM Automate 命令**	
exportSnapshot	validateConsolidationMetadata

Narrative Reporting 命令

适用于 **Narrative Reporting 的 EPM Automate 命令**		
addUsers	getIdleSessionTimeout	restoreBackup
addUsersToGroup	getIPAllowlist	roleAssignmentAuditReport
addUserToGroups	getRestrictedDataAccess	roleAssignmentReport
assignRole	getVirusScanOnFileUploads	runDailyMaintenance
cloneEnvironment	groupAssignmentAuditReport	sendMail
copyFileFromInstance	help	setDailyMaintenanceStartTime
copyFromObjectStorage	importLibraryArtifact	setEncryptionKey
copyFromSFTP	invalidLoginReport	setIdleSessionTimeout
copyToObjectStorage	listBackups	setIPAllowlist
copyToSFTP	listFiles	setManualDataAccess
createGroups	login	setRestrictedDataAccess
createNRSnapshot	logout	setVirusScanOnFileUploads
deleteFile	provisionReport	skipUpdate
deleteGroups	recreate	unassignRole
downloadFile	removeUserFromGroups	updateUsers
encrypt	removeUsers	upgrade
executeBurstDefinition	removeUsersFromGroup	uploadFile
exportLibraryArtifact	replay	userAuditReport
feedback	resetService	userGroupReport
getDailyMaintenanceStartTime		

Oracle Enterprise Data Management Cloud 命令

Oracle Fusion Cloud Enterprise Data Management 的 EPM Automate 命令		
addUsers	getIdleSessionTimeout	replay
addUsersToGroup	getIPAllowlist	resetService
addUserToGroups	getRestrictedDataAccess	restoreBackup
assignRole	getVirusScanOnFileUploads	roleAssignmentAuditReport
cloneEnvironment	groupAssignmentAuditReport	roleAssignmentReport
copyFileFromInstance	help	runDailyMaintenance
copyFromObjectStorage	importDimension	sendMail
copyFromSFTP	importSnapshot	setDailyMaintenanceStartTime
copySnapshotFromInstance	invalidLoginReport	setEncryptionKey
copyToObjectStorage	listBackups	setIdleSessionTimeout
copyToSFTP	listFiles	setIPAllowlist
createGroups	loadDimensionViewpoint	setRestrictedDataAccess
deleteFile	loadViewpoint	setManualDataAccess
deleteGroups	login	setVirusScanOnFileUploads
downloadFile	logout	skipUpdate
encrypt	provisionReport	unassignRole
exportDimension	recreate	updateUsers
exportDimensionMapping	removeUserFromGroups	upgrade
exportSnapshot	removeUsers	uploadFile
extractDimension	removeUsersFromGroup	userAuditReport
extractPackage	renameSnapshot	userGroupReport
feedback		
getDailyMaintenanceStartTime		

Planning、Planning 模块、自由形式、战略性人员规划和销售规划命令

Planning、Planning 模块、自由形式、战略性人员规划和销售规划的 EPM Automate 命令		
addUsers	getApplicationAdminMode	replay
addUsersToGroup	getDailyMaintenanceStartTime	resetService
addUserToGroups	getEssbaseQryGovExecTime	restoreBackup
applicationAdminMode	getIdleSessionTimeout	restructureCube
assignRole	getIPAllowlist	roleAssignmentAuditReport
autoPredict * 参阅脚注	getRestrictedDataAccess	roleAssignmentReport
clearCube	getSubstVar	runBatch
cloneEnvironment	getVirusScanOnFileUploads	runBusinessRule
compactCube	groupAssignmentAuditReport	runDailyMaintenance
copyFileFromInstance	help	runDataRule
copyFromObjectStorage	importAppAudit	runDMReport
copyFromSFTP	importAppSecurity	runIntegration
copySnapshotFromInstance	importCellLevelSecurity	runPipeline
copyToObjectStorage	importData	runPlanTypeMap
copyToSFTP	importDataManagement	runRuleSet
createGroups	importJobConsole	sendMail
deleteFile	importMapping	setApplicationAdminMode
deleteGroups	importMetadata	setDailyMaintenanceStartTime
dismissIPMInsights**	importSnapshot	setDemoDates
downloadFile	importValidIntersections	setEncryptionKey
enableQueryTracking	invalidLoginReport	setEssbaseQryGovExecTime
encrypt	listBackups	setIdleSessionTimeout
essbaseBlockAnalysisReport	listFiles	setIPAllowlist
executeAggregationProcess	login	setManualDataAccess
executeReportBurstingDefinition	logout	setRestrictedDataAccess
exportAppAudit	maskData	setSubstVars
exportAppSecurity	mergeDataSlices	setVirusScanOnFileUploads
exportCellLevelSecurity	provisionReport	simulateConcurrentUsage
exportData	recreate	skipUpdate
exportDataManagement	refreshCube	snapshotCompareReport
exportEssbaseData	removeUserFromGroups	sortMember
exportJobConsole	removeUsers	unassignRole
exportLibraryDocument	removeUsersFromGroup	updateGuidedLearningSettings
exportMapping	renameSnapshot	updateUsers
exportMetadata		upgrade
exportSnapshot		uploadFile
exportValidIntersections		userAuditReport
feedback		userGroupReport

Planning、Planning 模块、自由形式、战略性人员规划和销售规划的 EPM Automate 命令

* 自由形式、战略性人员规划和销售规划不支持此命令。
** 自由形式不支持此命令。

Profitability and Cost Management 命令

适用于 Profitability and Cost Management 的 EPM Automate 命令		
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
copyFromSFTP	importSnapshot	runDataRule
copyPOV	importTemplate	runDMReport
copySnapshotFromInstance	invalidLoginReport	runIntegration
copyToObjectStorage	listBackups	sendMail
copyToSFTP	listFiles	setDailyMaintenanceStartTime
createGroups	loadData	setEncryptionKey
deleteFile	loadDimData	setEssbaseQryGovExecTime
deleteGroups	login	setIdleSessionTimeout
deletePOV	logout	setIPAllowlist
deployCube	mergeSlices	setRestrictedDataAccess
downloadFile	optimizeASOCube	setManualDataAccess
enableApp	programDocumentationReport	skipUpdate
encrypt	provisionReport	setVirusScanOnFileUploads
exportDataManagement	recreate	unassignRole
exportMapping	removeUserFromGroups	updateUsers
exportQueryResults	removeUsers	upgrade
exportSnapshot	removeUsersFromGroup	uploadFile
exportTemplate		userAuditReport
feedback		userGroupReport
getDailyMaintenanceStartTime		

Enterprise Profitability and Cost Management 命令

适用于 **Enterprise Profitability and Cost Management** 的 **EPM Automate** 命令		
addUsers	exportTaskManagerAccessControl	removeUsers
addUsersToGroup	exportValidIntersections	removeUsersFromGroup
addUserToGroups	feedback	renameSnapshot
applicationAdminMode	getApplicationAdminMode	replay
assignRole	getDailyMaintenanceStartTime	resetService
calculateModel	getEssbaseQryGovExecTime	restoreBackup
clearCube	getIdleSessionTimeout	roleAssignmentAuditReport
compactCube	getIPAllowlist	roleAssignmentReport
copyDataByPointOfView	getRestrictedDataAccess	runBatch
cloneEnvironment	getSubstVar	runBusinessRule
copyFileFromInstance	getVirusScanOnFileUploads	runDailyMaintenance
copyFromObjectStorage	groupAssignmentAuditReport	runDataRule
copyFromSFTP	help	runDMReport
clearDataByPointOfView	importAppAudit	runIntegration
copySnapshotFromInstance	importAppSecurity	runPipeline
copyToObjectStorage	importCellLevelSecurity	sendMail
copyToSFTP	importData	runRuleSet
createGroups	importDataManagement	runTaskManagerReport
deleteFile	importJobConsole	setApplicationAdminMode
deleteGroups	importMapping	setDailyMaintenanceStartTime
downloadFile	importMetadata	setDemoDates
deletePointOfView	importSnapshot	setEncryptionKey
deployTaskManagerTemplate	importValidIntersections	setEssbaseQryGovExecTime
enableQueryTracking	invalidLoginReport	setIdleSessionTimeout
encrypt	listBackups	setIPAllowlist
executeAggregationProcess	listFiles	setManualDataAccess
executeReportBurstingDefinition	login	setRestrictedDataAccess
exportAppAudit	logout	setSubstVars
exportAppSecurity	maskData	setVirusScanOnFileUploads
exportCellLevelSecurity	mergeDataSlices	skipUpdate
exportData	provisionReport	snapshotCompareReport
exportDataManagement	recreate	sortMember
exportEssbaseData	refreshCube	unassignRole
exportJobConsole	removeUserFromGroups	updateGuidedLearningSettings
exportLibraryDocument		updateUsers
exportMapping		upgrade
exportMetadata		uploadFile
exportMetadata		userAuditReport
exportSnapshot		userGroupReport
		validateModel

Tax Reporting 命令

适用于 Tax Reporting 的 EPM Automate 命令		
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
copyFromSFTP	importCellLevelSecurity	runDMReport
copyOwnershipDataToNextYear	importData	runIntegration
copySnapshotFromInstance	importDataManagement	runPipeline
copyToObjectStorage	importJobConsole	runRuleSet
copyToSFTP	importMapping	runSupplementalDataReport
createGroups	importMetadata	runTaskManagerReport
deleteFile	importOwnershipData	sendMail
deleteGroups	importSnapshot	setApplicationAdminMode
deployFormTemplates	importSupplementalCollectionData	setDailyMaintenanceStartTime
deployTaskManagerTemplate	importSupplementalData	setDemoDates
downloadFile	importValidIntersections	setEncryptionKey
encrypt	invalidLoginReport	setEssbaseQryGovExecTime
essbaseBlockAnalysisReport	listBackups	setIdleSessionTimeout
executeReportBurstingDefinition	listFiles	setIPAllowlist
exportAppAudit	login	setManualDataAccess
exportCellLevelSecurity	logout	setRestrictedDataAccess
exportData	maskData	setSubstVars
exportDataManagement	provisionReport	setVirusScanOnFileUploads
exportEssbaseData	recomputeOwnershipData	simulateConcurrentUsage
exportJobConsole	recreate	skipUpdate
exportLibraryDocument	refreshCube	snapshotCompareReport
exportMapping	removeUserFromGroups	unassignRole
exportMetadata	removeUsers	updateGuidedLearningSettings
exportOwnershipData	removeUsersFromGroup	upgrade
exportSnapshot	removeUsersFromTeam	updateUsers
exportTaskManagerAccessControl	renameSnapshot	uploadFile
exportValidIntersections		userAuditReport
feedback		userGroupReport
getApplicationAdminMode		