Anaplan Connect Guide

Version 4.3.4

(Last updated June 30, 2025)

Table of Contents

What's New in Anaplan Connect 4.3.4

There’s a note about -maxretry count on pages 19, 23, 70, and 72:
When you have a -put parameter, the -maxretry count should be run before the -put command.
Also use -maxretry directly after the -models or -model_id command.

If you use -maxretry count in conjunction -httptimeout and -retrytimeout, use these commands in alphabetical order.

This version contains security updates and an expansion of Java compatibility to version 21.

There’s a note for Windows users on page 10:

“For Windows users, if you receive an "Invalid key path" error when you execute your script, make sure that your Anaplan Connect folder is installed in the same drive as your system root user.”

Introduction

Anaplan Connect is an API Client with a command-line interface that supports the following types of Anaplan operations:

- Import, Export, Process, Delete operations leveraging Model Actions
- Data import and export using Transactional APIs
- Model metadata query using Transactional APIs

The advantages include:

- Simple download and set-up process
- Provides users with a lightweight tool to automate integration tasks
- No need for manual work in the Anaplan GUI for each run. You can schedule jobs to run automatically at the interval you want.
- Support for CA, certificate-based authentication for enhanced security.

Create Shell Scripts for Integrations

You write Anaplan Connect integrations with shell scripts, BAT files for Windows, and Bash files for Linux/Unix. This document provides examples for Windows, Linux, and Mac OS operating systems. Anaplan Connect is designed to support automated and ad-hoc jobs. The examples should work as either scheduled jobs, run from a server or workstation, or as unscheduled jobs to be run as needed.

Example: A batch file named RunMyImport.bat that loads a text file, Europe.txt, onto the Anaplan server. If you also use a scheduling tool, the batch file can run itself at any hour and interval you choose (daily, weekly) without you having to log onto Anaplan or be present.

Anaplan Connect compresses the files during upload. Do not refer to zip files in the .bat or .sh file, this is not supported.

Prerequisites

Minimum system requirements

Disk space

The disk space you need depends on the sizes of the files to import or export from Anaplan. As a general rule, disk space should be at least twice the size of your data payloads. This ensures that there is enough disk space for data to chunk. The need for disk space increases greatly when you store intermediate files during the integration process.

Dynamic memory

The amount of dynamic memory you need depends on the Java version you use. Also consider the maximum memory used by your Anaplan Connect scripts. Rule of thumb: the maximum size of data buffered is driven by the chunk size. A chunk ranges is between 1 MB and 50 MB.

Central processing unit (CPU)

The CPU you need for Anaplan Connect ties directly to data payloads. Ask yourself if you typically run multiple Anaplan Connect integrations in parallel. A modern CPU can handle multiple, simultaneous integration jobs. However, if your data payloads are particularly large or complex, you may need a more powerful CPU.

CPU guidelines

The minimum system requirements for Anaplan Connect depend on your specific use case and the size and complexity of the data payloads from your scripts. We recommend:

- A modern CPU with multiple cores
- At least 8GB of RAM
- Sufficient disk space for your data payloads ○ Space for any intermediate files you generate throughout the integrations.

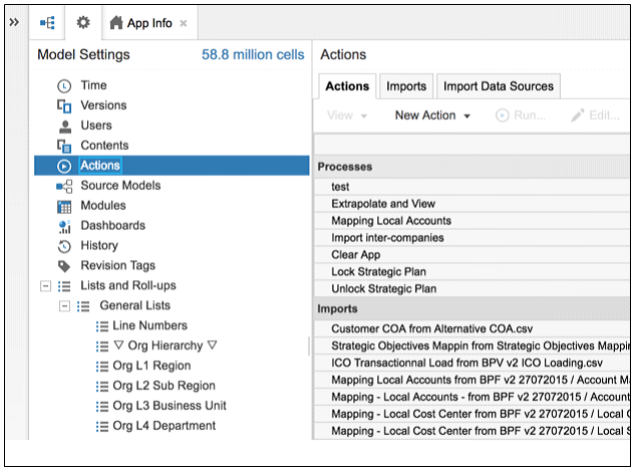
Firewall and Proxy Access

See Anaplan's current Domain and IP ranges on our support page. Add api.anaplan.com to your IP allow list. You might also refer to this URL information support page.

- Ensure that outbound connectivity to these URLs is available. Anaplan Connect reaches your models via HTTPS.

Access to Anaplan model with the actions configured

Open an Anaplan model that already has the actions (Import, Export, Delete, or Process) that you want Anaplan Connect to run.



If you do not have access to Anaplan, work with someone who has the ability to create actions in Anaplan.

Java version and third-party data sources

- If you want to use Anaplan Connect to import from an ODBC data source, note that Java 8 does not support the JDBC-ODBC Bridge (see https://blogs.oracle.com/Lance/entry/removal_ of_the_jdbc_odbc).
- Anaplan Connect supports Java Database Connectivity (JDBC), which means it is possible to work with many third-party data sources.
    - JDBC for Oracle, Access, Excel
    - Import through a JDBC Connection for a Microsoft SQL Server database

Note: **Supported Oracle Java and OpenJDK versions:**

We have successfully tested Anaplan Connect with following JAVA versions 8, 11, 17 and 21. And specifically:

- Oracle JAVA JDK 1.8.0_342 , 11.0.16 and 17.0.5
- Azul OpenJDK 1.8.0_ 232-b18 and Azul OpenJDK 8.0.242-b2

If you're using an SSO-enabled workspace

If the actions you want Anaplan Connect to run are for models in a workspace using single sign-on, we recommend using Certificate Authority (CA) Authentication.

If you use Anaplan's Basic authentication, the Anaplan Connect Single Sign-on (SSO) user must be an Exception User. An Exception User can authenticate by username and password or by certificate, rather than through SAML.

See: https://help.anaplan.com/anapedia/Content/Administration_and_Security/Security/Single_Sign-on.html.

Download and Setup

1. Go to the Downloads page in Anapedia.
2. Expand **Anaplan Connect (API Client)**.
3. Click the Anaplan Connect client version you want and save the zip file to your hard drive.
4. Extract the zip to a directory.

Note: If the folder name contains parentheses, an error might occur when using Anaplan Connect. Do not install Anaplan Connect to a folder that has parentheses in its name. For example, Program Files (x86).

Install Bouncy Castle Library

If you use private keys directly as part of your CA certificate authentication, Anaplan Connect 1.4.3 and later requires the Bouncy Castle library in your Java installation directory. To install Bouncy Castle, you can either manually copy the jar file included with the Anaplan Connect API client or run associated scripts. You must have administrative or root permissions on your system to install Bouncy Castle.

Manually copy the jar file

To manually copy the jar file to your Java library:

1. Log in as an administrator to your system (if you do not already have administrator permissions).
2. In your Anaplan Connect installation, navigate to the /lib subfolder and find the bcprov-jdk15on-164.jar file.
3. Copy the bcprov-jdk15on-164.jar file to $JAVA_HOME/jre/lib/ext on the machine running Anaplan Connect. This step requires administrator permissions.

Note: $JAVA_HOME is an environment variable in your system that defines the location of your Java installation.

Run a script to install the Bouncy Castle library

If you prefer not to manually copy the .jar file into your Java libraries, we have included scripts for Windows and Mac/Linux to perform this task. These scripts must be run by someone with administrator or root access.

Note: Before running these scripts, ensure the directory defined in your system's JAVA_HOME environment variable does not end with a slash. The scripts will not run properly if your JAVA_HOME variable directory ends with a slash.

Windows

1. Navigate to the /lib subdirectory in your Anaplan Connect installation.
2. Right-click InstallBouncyCastle.bat and run as administrator.

Mac/Linux

3. In a terminal window, navigate to the /lib subdirectory in your Anaplan Connect installation.
4. As an administrator, assign executable permissions to the InstallBouncyCastle.sh script. For example, as sudo, run chmod +x InstallBouncyCastle.sh.
5. Run InstallBouncyCastle.sh.

Create scripts for Anaplan Connect (Basic authentication)

Anaplan Connect enables you to write scripts to easily apply Anaplan APIs for a variety of tasks.

You can use the script below to verify that you are able to create and use new Anaplan Connect jobs. This script uses basic authentication, so you will be prompted for your Anaplan password before the job is run.
To follow along with the example, you must first log into Anaplan Connect and create an Import Action. Anaplan Connect supports both names and ID numbers for workspaces and models, within integration scripts.

Windows

Save and run the following script in the same directory as AnaplanClient.bat. The script is also available in the examples folder at examples\example.bat. The script contains both the user and associated password, but the password is optional. Leaving it off will cause Anaplan Connect to prompt for the password prior to executing the script.

Note: You cannot use exclamation marks (!) in passwords with Windows script files. This is a factor in Windows scripts themselves, not with Anaplan Connect.

```
@echo off rem This example loads a source text file and runs an Anaplan import
into a module. rem For details of how to configure this script see doc\Anaplan
Connect User Guide.doc set AnaplanUser="your.email@company.com:password" set
WorkspaceId="My Workspace" set ModelId="My Model" set Operation=-service
```

```
"https://api.anaplan.com" -auth "https://us1a.app.anaplan.com"
-file "My Source.txt" -put "C:\My Source.txt" -import "My Module from My Source.txt" -execute
  -output "C:\My Errors.txt" rem *** End of settings - Do not edit below this line *** setlocal
enableextensions enabledelayedexpansion || exit /b 1 cd %~dp0 if not %AnaplanUser% == "" set
Credentials=-user %AnaplanUser% set Command=.\AnaplanClient.bat %Credentials% -workspace
%WorkspaceId% -model %ModelId% %Operation%
@echo  %Command%
cmd /c %Command%
pause
```

Where:

your.email@company.com:password	your Anaplan login credentials
My Workspace	Name or ID of your workspace
My Model	Name or ID of your model
My Source.txt	a flat file in your local host
C:\My Source.txt	full path to the flat file in your local host
C:\My Errors.txt	full path to where you want Anaplan Connect to create a dump file in your local host. The dump file contains the records that could not be imported.

Linux or Mac OS

Save and run the following script in the same directory as AnaplanClient.sh. The script contains both the user and password, but the password is optional. Leaving it off will cause Anaplan Connect to prompt for the password prior to executing the script.

```
#!/bin/sh
#This example uploads a file and runs an import
AnaplanUser="your.email@company.com:password"
WorkspaceId="workspaceid"
ModelId="modelid"
```

```
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com

" ImportName="import action name"
FileName="import data source name"
FilePath="import file location"
ChunkSize=1
ErrorDump="error dump location"
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}' -execute -output
'${ErrorDump}' "
#_____________________________ Do not edit below this line _____________________________ if
[ "${AnaplanUser}" ]; then
  Credentials="-user
${AnaplanUser}" fi if [
"${CertPath}" ]; then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"       # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi echo cd "`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c

"${Command}"
```

Where:

your.email@company.com:password	your Anaplan login credentials
WorkspaceID	Name or ID of your workspace
ModelID	Name or ID of your model

import action name	name of the import Action you created
import data source name	a flat file in your local host
import file location	full path to the flat file in your local host
error dump location	full path to where you want Anaplan Connect to create a dump file in your local host. The dump file contains the records that could not be imported.

Create scripts for Anaplan Connect (OAuth 2.0)

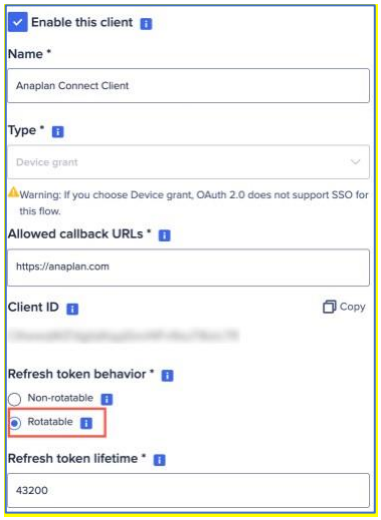
In addition to Basic and Certification-based authentication, Anaplan Connect also supports OAuth 2.0. It is an open standard for authorization that enables apps to access data without the need for confidential login information.

- OAuth 2.0 works over HTTPs and authorizes devices, APIs, and servers, with access tokens.
- There are three new optional Anaplan Connect parameters: `--forceRegister`, `--oauthclient-id` and `--rotatable`.
- You can create an OAuth client in Anaplan, and apply it in your Anaplan scripts.

Note: Only use Device grant when you create your OAuth 2.0 client.

Note: Keep in mind that if you set up your OAuth with the rotatable token, you must include the parameter '--rotatable' in your script.

Enable client dialog:



See Anaplan's OAuth 2.0 client for background information.

OAuth Device Grant Flow

To use an Anaplan Connect script with OAuth, the particular device that hosts the script needs to be authorized. To authorize the device, an end user needs to interactively execute a script with the parameter `--forceRegister`. This parameter will provide a user with a unique URL where the user will authenticate with Anaplan. Once the device grant flow is complete, then the `--forceRegister` can be removed and the script can be scheduled.

To begin, there are two sample scripts in the "Examples" folder where Anaplan Connect. They are named:

Authentication - Generate OAuth Token With Force Register.bat (Windows)

Linux or Mac OS: Authentication - Generate OAuth Token With Force Register.sh (Linux and Mac OS)

Depending on the operating system, edit the script and update it with your particular **OAuth Client ID** created when the OAuth client was created.

Authentication - Generate OAuth Token With Force Register.bat

```
@echo off
rem For Windows OS : This script will store the refresh token for Oauth 2.
rem This script should be execute only one time before Oauth 2 is used in scripts.
rem After you run this script you will use the client id to be authenticate
rem For details of how to configure this script visit https://help.anaplan.com/anapedia/Content/Downloads/DL Do

set ClientId="The client id"
set ServiceUrl="https://api.anaplan.com"
set AuthUrl="https://auth.anaplan.com"


set Operation=-debug -service %ServiceUrl% -auth %AuthUrl% --forceRegister -W

rem *** End of settings - Do not edit below this line ***

setlocal enableextensions enabledelayedexpansion || exit /b 1
cd %~dp0
if not %ClientId% == "" set Credentials=-oauth %ClientId%
set Command=..\AnaplanClient.bat %Credentials%  %Operation%
@echo %Command%
cmd /c %Command%
pause
```

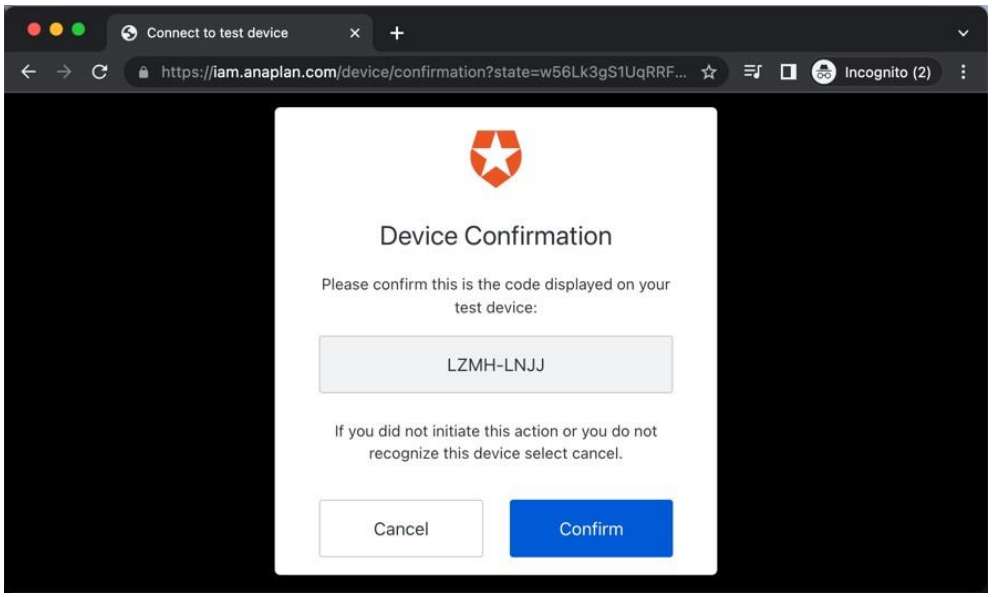
From the command line or terminal, interactively execute the script and notice that a URL will be outputted to the console log:

```
2023-01-09 15:30:27 DEBUG [c.a.client.Program  :2275 ] 33589 |--   ========================================================================
2023-01-09 15:30:27 DEBUG [c.a.client.Program  :2276 ] 33589 |--   Anaplan Connect 4.0.1
2023-01-09 15:30:27 DEBUG [c.a.client.Program  :2278 ] 33589 |--   Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)/ (25.351-b10)/ 1.8.0_351
2023-01-09 15:30:27 DEBUG [c.a.client.Program  :2281 ] 33589 |--   (Mac OS Xx86_64)/13.1
2023-01-09 15:30:27 DEBUG [c.a.client.Program  :2284 ] 33589 |--   ========================================================================
2023-01-09 15:30:28 INFO  [c.a.client.Service  :95   ] 33589 |--   Initializing Service...
2023-01-09 15:30:28 INFO  [.DeviceAuthenticator:60   ] 33589 |--   Authenticating via Device...
2023-01-09 15:30:28 INFO  [.DeviceAuthenticator:77   ] 33589 |--   Deleting already existing JKS and re-registering the device...
2023-01-09 15:30:28 INFO  [.DeviceAuthenticator:166  ] 33589 |--   Registering device...
2023-01-09 15:30:28 INFO  [.DeviceAuthenticator:128  ] 33589 |--   Please activate device using following url: https://iam.anaplan.com/activate?user_code=PHTB-FVJC
2023-01-09 15:30:28 INFO  [.DeviceAuthenticator:129  ] 33589 |--   User code: PHTB-FVJC
2023-01-09 15:30:29 ERROR [.DeviceAuthenticator:148  ] 33589 |--   1 attempt. User has yet to authorize the device code in browser.
2023-01-09 15:30:32 ERROR [.DeviceAuthenticator:148  ] 33589 |--   2 attempt. User has yet to authorize the device code in browser.
2023-01-09 15:30:35 ERROR [.DeviceAuthenticator:148  ] 33589 |--   3 attempt. User has yet to authorize the device code in browser.
```

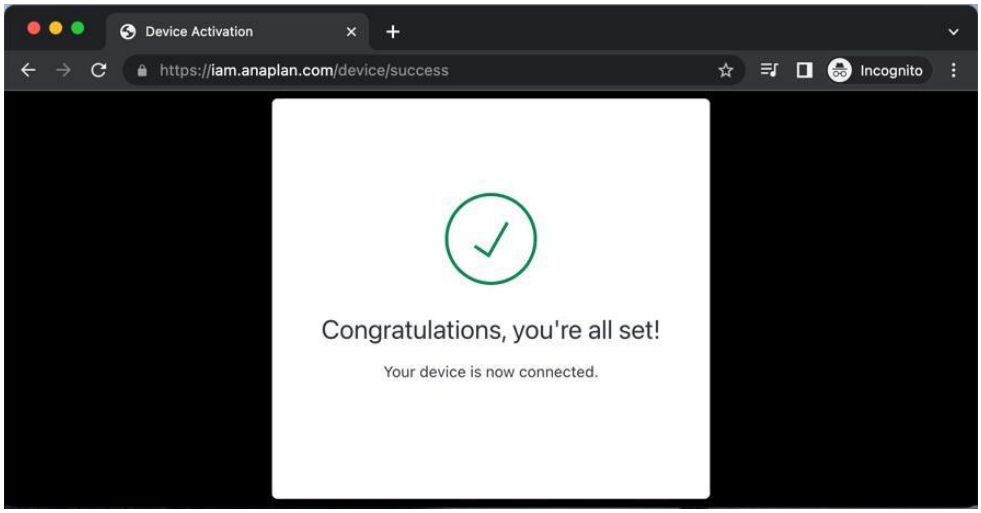
Copy the URL to your clipboard and open the link in a new “Incognito” browser window.

Notes:
The URL expires after 30 seconds.
For Windows users, if you receive an "Invalid key path" error when you execute your script, make sure that your Anaplan Connect folder is installed in the same drive as your system root user.

When the URL loads, login into Anaplan with the associated user credentials. Upon successful login the following similar page will appear. Click **Confirm**.



If successful, the following page should appear:



Now that the particular device has been authorized with a specific valid **OAuth Client ID**, any Anaplan Connect script using the particular Client ID will be able authenticate with Anaplan and execute.

Note: For sample scripts for each authentication method, see Appendix G: Linux and macOS Scripts.

Configure Your Scripts

Every script contains the following information:

User credentials or certificate information.

Workspace name or ID - identifies your Anaplan workspace (always unique) l Model ID - identifies your Anaplan model (always unique) l One or more actions from your Anaplan model, such as a specific import or export

When running a script at an interactive terminal, the user is prompted for the password if it is not included in the script.

Jobs can take longer than the 30 minutes allocated to an Auth token, Anaplan Connect allows calls to be made to the API after this expiry limit without re-entering the authentication details.

Warning: A script that runs without user interaction must contain the credentials. Take measures to secure the file, machine, and account.

Locate the Workspace ID and Model ID (Optional)

The connector supports both names and ID numbers for workspaces and models. You can copy the name of workspace or model from Anaplan UI. To find your model ID:

1. Log in to Anaplan.
2. Open your model.
3. In the upper-right corner, click Help > About. The About dialog displays the values of the workspace and your model ID.
4. Copy the workspace ID to the line of your batch file that begins with set WorkspaceId=".
5. Copy the model ID to the line of your batch file that begins with set ModelId=".

Locate the Actions for Anaplan Connect to Perform

1. Open the Anaplan model that has the actions you want Anaplan Connect to perform.
2. In **Model Settings > Actions**, note the exact names of the actions, including capitalization and file extensions. For additional help, read the Anapedia articles.

Certificate Authentication

Starting with Anaplan Connect 1.4, Certificate Authority (CA) authentication lets you use certificates from a supported root CA, or Intermediate Certificate ending in a supported root CA. Use your certificate for CA Certificate based authentication with Anaplan APIs and Anaplan Connect. For more information, see Administration: Certificates in Anapedia.

Storing sensitive pieces of information like your certificate private keys in the filesystem can be insecure. It's good practice to use a Java KeyStore (JKS) to protect the private key and certificate with a KeyStore password. Anaplan Connect can work with such KeyStores if they have been created for Anaplan Connect to consume.

Note: You must register your public certificate in Anaplan before using these certificates in integrations. For more information, see Manage your Certificates in Anapedia.

You need the following command-line tools for this procedure:

Tool	Location
openssl	https://www.openssl.org/source/ Note: This link takes you outside of Anaplan to download openssl. For more information on openssl usage examples, review articles on Anaplan Community. Anaplan does not support openssl and does not assume any responsibility for issues arising out of downloading and using openssl.
keytool	keytool comes bundled with your JAVA distribution

When you obtain a certificate from your CA (Certificate Authority), the certificate is usually issued as two files: a public certificate (public cert) and a private key. The public cert and private key may also be issued in a single file by your CA. Contact your CA or internal IT team for details and for information on the process to obtain a certificate.

If you obtain both the public cert and the private key in a single file, use openssl to extract and save both files separately in PEM format.

In the following example, a single PKCS12 formatted file was issued containing both a public cert and private key.

To extract a public kert: `openssl pkcs12 -in Client_certificate.p12 -`

`nokeys -out CERTIFICATE.pem`

You can use the following commands to extract your private key. Instructions for encrypted and unencrypted private keys are included. We recommend that you use the encrypted option.

Note: The exact command varies depending on the type and format of your certificate. Refer to the openssl documentation for more details and compare against the file type you receive.

Encrypted Private Key

When you extract the private key with this command, you're prompted to create a passphrase. Record your passphrase in a safe place, as you need it to use the private key in your authentication.

```
openssl pkcs12 -in Client_certificate.p12 -nocerts -out PRIVATE_ KEY.pem

Convert your encrypted private key to an encrypted pkcs8 file in PEM format.

openssl pkcs8 -inform PEM -in PRIVATE_KEY.pem -outform PEM -out AC_PRIVATE_KEY.pem -passout pass:
YourPassphrase
```

Unencrypted Private Key

To extract the private key without being prompted to create a passphrase: `openssl`

```
pkcs12 -in Client_certificate.p12 -nocerts -out PRIVATE_ KEY.pem -nodes
```

Your Tenant Administrator must then register the modified public cert in Anaplan. This step is essential to using the certificate in integrations.

Create your authentication method. You can either use a private key or create a Java KeyStore.

Option 1: Use a Private Key with Anaplan Connect

Use your private key with Anaplan Connect by providing to the *-privateKey* argument with your passphrase.

Depending on how your private key was created, it may be encrypted with a password. Encrypted keys will prompt for a password when used, so the password must be included in your Anaplan Connect scripts.

- Encrypted Key: Set the password at the end of the private key's file path. The script below uses the 'Passphrase' parameter to obtain the password and then adds it to the end of the private key's ('PrivateKey') file path.
- Unencrypted Key: There is no password to include with the key, but you must keep the colon ':' delimiter at the end of the private key's file path. If you do include a password, an error occurs. To use the example script below, leave the 'Passphrase' field empty.

Example Run script

```
#!/bin/sh

This example uploads a file and runs an import

WorkspaceId="<workspace ID>"

ModelId="<model ID>"

PrivateKey="<path/to/privatekey.pem>"

Passphrase="<passphrase>"

FilePath="<path/for/file import>"

Certificate="<path/to/certificate>"

Operation="-debug -service 'https://api.anaplan.com' -auth 'https://us1a.app.anaplan.com' -workspace
${WorkspaceId}
-model ${ModelId} -chunksize 1 -file 'Employee Planning.csv' -put ${FilePath} -import 'Employee Planning
from

Employee Planning.csv' -execute -output 'p2-dump/Employee Planning from Employee Planning.csv' "
#_____________________________ Do not edit below this line

_____________________________ Credentials="-certificate ${Certificate} privatekey

${PrivateKey}:${Passphrase}" echo cd "dirname "$0"" cd "dirname "$0"" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same directory

as AnaplanClient.sh." >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please

ensure you have executable permissions on AnaplanClient.sh." >&2 exit

1 fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"

exec /bin/sh -c

"${Command}"
```

Option 2: Create a Java KeyStore

For a video example of creating a Java KeyStore, see:
https://community.anaplan.com/t5/OnDemandCourses/Certificate-Authentication-Process/ta-p/74752

1. Create a PKCS12 bundle called "keystore bundle.p12" from the Private-Key and the CA Certificate.

   Make sure you use a suitable keystore alias to uniquely identify the entry as this name is required for the keystorealias argument when you run Anaplan Connect.

   ```
   $ openssl pkcs12 -export -in <CERTIFICATE.pem> -inkey <PRIVATE_

   KEY.pem> -out keystore_bundle.p12 -name

   <KEYSTORE_ALIAS> -CAfile <CERTIFICATE.pem> -caname root
   ```

2. You are prompted for your Keystore password. You will use the same password for creating a Java keystore in the next step.

3. Use the PKCS12 bundle to create the JKS keystore called "my_keystore.jks".

```
$ keytool -importkeystore -deststorepass <KEYSTORE-PASSWORD> destkeystore my_keystore.jks

-srckeystore keystore_bundle.p12 -srcstoretype PKCS12
```

You are prompted for your Keystore password. Use the same password that was used to create the pkcs12 bundle in the previous step. The .jks file contains your KeyStore that securely holds the PrivateKey and the CA Certificate.

KeyStore Wizard for Anaplan Connect 1.4

Alternatively, you can use the KeyStore Wizard tool in Windows to create a Java Keystore:

https://community.anaplan.com/t5/Knowledge/KeyStore-Wizard-for-Anaplan-Connect-1-4/ta-p/38831#U38831

Note: The KeyStore Wizard tool is not supported by Anaplan. Anaplan does not assume any responsibility for issues arising from its download and use.

Use Keystore with Anaplan Connect

Use the Keystore with Anaplan Connect by providing to the -keystore, -keystorepass and -keystorealias arguments with values used when building the Keystore.

Example Run script

```
#!/bin/sh
# This example runs a Keystore with Anaplan Connect
CACertPath="</path/to/CA.crt>"
KeyStorePath="</path/to/keystore.jks>"
KeyStorePass="<your_password>"
KeyStoreAlias="<your_test_alias>"
WorkspaceId="<workspace_ID>"
Operation="-debug -service 'https://api.anaplan.com' -auth 'https://us1a.app.anaplan.com' -w
${WorkspaceId} -M"
#_________________ Do not edit below this line
___________________ if [ "${CACertPath}" ]; then
  Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
  #Credentials="-certificate ${CACertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
  -keystorealias                    # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE
${KeyStoreAlias}" certificate VIA - fi echo
cd "`dirname "$0"`" cd
```

```
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then

echo "Please ensure this script is in the same directory as AnaplanClient.sh." >&2


  exit 1 elif [ ! -x AnaplanClient.sh
]; then echo "Please ensure you have
executable      permissions      on
AnaplanClient.sh."

  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} ${Operation}"
/bin/echo "${Command}"
exec     /bin/sh     -c
"${Command}"
```

Use Anaplan Bulk API capabilities to run file-based integrations

You can run export, import, process and delete actions within Anaplan Connect. These actions require you to pre-configure actions within your Anaplan models.

Upload and Download Files

Make sure the import file has only one type of column separator as Anaplan Connect supports only one when you upload a file to Anaplan. Files are uploaded to a location on a customer-hosted machine, where the import script can access them before they are imported into Anaplan.

The -chunksize parameter allows you to configure the upload chunk size parameter between 1 and 50MB to accommodate large file uploads.

Example script for upload

Linux or Mac OS:

```
#!/bin/sh

#This example uploads a file and runs an import
AnaplanUser="user@anaplan.com"

AnaplanPassword="Password" set

WorkspaceId="8a1234567897c12b014bf01234567890"

set ModelId="CD1234D60CA84E9A123C1C5D061C1234"

Operation="-debug -service 'https://api.anaplan.com' -auth 'https://us1a.app.anaplan.com' -file

'file_to_ upload.csv' -chunksize 1 -put 'file_to_upload.csv' -output 'errors.txt'"
#_____________________________ Do not edit below this line
_____________________________ if [ "${AnaplanUser}" ]; then

  Credentials="-user

${AnaplanUser}:${AnaplanPassword}" fi echo cd

"`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same

  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you

have executable permissions on AnaplanClient.sh."
```

```
  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} -workspace ${WorkspaceId} -model ${ModelId} ${Operation}"
/bin/echo "${Command}" exec /bin/sh -c

"${Command}"
```

Create an Import Script

Example script for Import

Note: If you use the -maxretry count and -retrytimeout parameters in your script, define them before the -execute parameter. When you have a -put parameter, the maxretry count should be placed before *-put*. For more information, see Appendix C: List of all Operation Commands and Appendix G: Linux and MacOS Scripts.

Import (basic authentication)

```
@echo off rem This example loads a source text file and runs an Anaplan import
into a module. rem For details of how to configure this script see doc\Anaplan
Connect User Guide.doc set AnaplanUser=anaplan.user@anaplan.com:Password set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CD1234D60CA84E9A123C1C5D061C1234"  set
Operation=-service "https://api.anaplan.com" -auth

"https://auth.anaplan.com"
-file "Employee.txt" -put "C:\AnaplanConnect\Import\Employee.txt"
-import "New Hire from Employee.txt"
 -execute -output "C:\My Errors.txt" rem *** End of settings

 - Do not edit below this line *** setlocal enableextensions enabledelayedexpansion || exit
/b 1 cd %~dp0 if not %AnaplanUser%
== ""  set Credentials=-user %AnaplanUser% set Command=.\AnaplanClient.bat %Credentials% -
workspace %WorkspaceId% -model %ModelId% %Operation%
@echo  %Command%
cmd /c %Command%
pause
```

Import (certificate authentication)

```
@echo off rem This example loads a source text file and runs an Anaplan import
into a module. rem For details of how to configure this script see doc\Anaplan
Connect User Guide.doc set CertPath="C:\certs\cert.pem" set
KeyStorePath="C:\certs\AC1.4_keystore.jks"  set KeyStorePass="keystorepass"
set KeyStoreAlias="keystorealias"  set
```

```
WorkspaceId="8a1234567897c12b014bf01234567890"  set
ModelId="CD1234D60CA84E9A123C1C5D061C1234"

set Operation=-service "https://api.anaplan.com" -auth

"https://auth.anaplan.com"

-file "Employee.txt" -put "C:\AnaplanConnect\Import\Employee.txt"
-import "New Hire from Employee.txt"

-execute -output "C:\My Errors.txt" rem


*** End of settings - Do not edit below this line *** setlocal enableextensions

enabledelayedexpansion || exit /b 1 cd %~dp0 set Credentials=-certificate %CertPath% -keystore

%KeyStorePath% -keystorepass %KeyStorePass% -keystorealias
%KeyStoreAlias% set Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model

%ModelId% %Operation% @echo

%Command% cmd

/c %Command%

pause
```

Set the Import Operation

As a best practice, the name of the Import action should indicate the name of the file (or other source) from which data will be imported, such as *Import From Employee.txt*.

Example:

```
set Operation=-service "https://api.anaplan.com" -auth

"https://us1a.app.anaplan.com"

-file "Employee.txt" -put "C:\AnaplanConnect\Import\Employee.txt"

-import "New Hire from Employee.txt" -execute -output

"C:\ImportDumpFilesDirectory"
```

where:

-file “Employee.txt”	indicates the action uses a file named Employee.txt. Note that .csv format is also supported.
-put “C:\AnaplanConnect\Import\Employee.txt"	upload the file to the specified absolute or relative path to the Anaplan Server.

-import “New Hire from Employee.txt”	an Import action with the specified name. To see this name in Anaplan, in **Settings** click **Actions** and view the list of Imports. As a best practice, name your import (or export) action such that it matches the name of the file.
-execute	runs the action
-output "C:\ImportDumpFilesDirectory"	generates a file at the specified directory that lists any Import errors. This is optional. This file is created only if at least one Import error occurs.

Linux and Mac OS example for Import

In Linux or Mac OS, use single-quotes instead of double quotes. In this example, compare the double-quotes of Windows with the single-quotes of Linux and Mac OS.

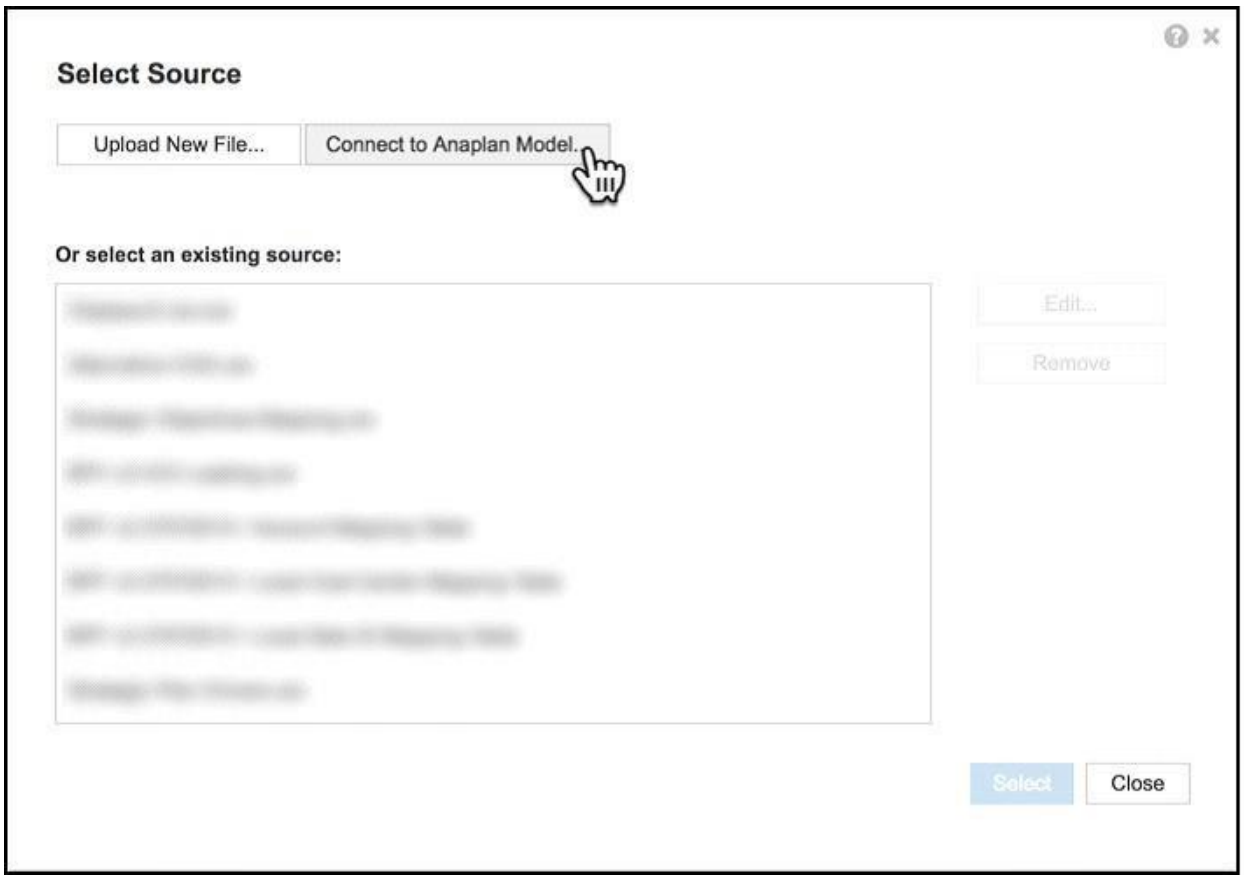
Windows	-put "C:\testdata\Europe P&L.txt" or: -p "C:\testdata\Europe P&L.txt"
Linux or Mac OS	-put '/Users/user1/testdata/Europe P&L.txt' or: –p '/Users/user1/testdata/Europe P&L.txt'

For Linux and Mac OS examples of import scripts, see Appendix G.

Model-to-model Import

This example runs a model-to-model import within Anaplan, transferring data from the **Installation Sales** module in Model2 to the **P&L** module in *MyBudgetModel*.

1. Run the import manually. On the Data menu, click Import then click **Connect to Anaplan Model**.



2. Select a module (or list) as the source of the import.

3. Click **Run Import** then edit and run the batch file.

   In the model, note the **Import ID** that is used in the batch file, which in the example below is *P&L from Model2 / Installation Sales*.

   There is a space before and after the forward slash "/" in the string 'P&L from Model2 / Installation Sales'.

```
@echo off rem This example runs a model to model import
within Anaplan. set
AnaplanUser="firstname.lastname@company.com" set
rem You can use names or IDs of workspaces & models
in the script set
WorkspaceId="8a819488459fa63301462b73fe785786" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-service
"https://api.anaplan.com" -auth "https://us1a.app.anaplan.com" -import "P&L from
Model2 / Installation Sales" -execute rem *** End of settings - Do not edit below
this line ***
```

Create an Export Script

Example Batch file for Export

Note: If you use the -maxretrycount and -retrytimeout parameters in your script, define them before the -execute parameter. When you have a -put parameter, the maxretry count should be placed before *-put*. For more information, see Appendix C: List of all Operation Commands and Appendix G: Linux and MacOS Scripts.

Export (basic authentication)

```
@echo off set AnaplanUser="Anaplan.User@anaplan.com:Password" set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelID="CD1234D60CA84E9A123C1C5D061C1234" set Operation=-service

"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-export "Employee by Department.xls" -execute -get "C:\Employee.xls" rem *** End of settings
- Do not edit below this line *** setlocal enableextensions enabledelayedexpansion || exit
/b 1 cd %~dp0 if not %AnaplanUser% == "" set Credentials=-user %AnaplanUser% set
Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model %ModelId%
%Operation% @echo %Command% cmd /c %Command% pause
```

Export (certificate authentication)

```
@echo off set CertPath="C:\certs\cert.pem" set
KeyStorePath="C:\certs\AC1.4_keystore.jks" set
KeyStorePass="keystorepass" set
KeyStoreAlias="keystorealias" set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelID="CD1234D60CA84E9A123C1C5D061C1234" set Operation=-service
"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-export "Employee by Department.xls" -execute -get "C:\Employee.xls" rem *** End of settings - Do not
edit below this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 cd %~dp0 set
Credentials=-certificate %CertPath% -keystore %KeyStorePath% -keystorepass %KeyStorePass%
keystorealias %KeyStoreAlias%



set Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model %ModelId% %Operation%
@echo
```

```
%Command% cmd
/c %Command%
pause
```

Set the Export Operation

Example:

```
set Operation=-export "Employee by Department.xls" -execute -get "C:\Employee.xls"
```

Where:

-export "Employee by Department.xls"	indicates that an Export action exists and is named "Employee by Department.xls"
-execute	runs the Export
-get "C:\Employee.xls"	creates a new file with the exported data at the specified path

An Export action cannot generate an error file, so we do not specify a path for it.

For Linux and Mac OS examples of export scripts, see Appendix G.

Using JDBC for Imports and Exports

Java Database Connectivity (JDBC) is a Java library for connecting to databases. Anaplan Connect can use it to import data from a database to Anaplan, as well as to export data from Anaplan to a database.

Import from a database using JDBC

Provided you have the appropriate JDBC driver, you can link directly into Anaplan from a compatible database. Both lists and module data can be imported into Anaplan in this way.

To use the JDBC import feature, your Anaplan Connect script should contain your model's info, as well as the path to your properties file. Your model's info includes the workspace ID and model ID.

Mac:

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
file '${FileName}' -jdbcproperties
'${path/to/JdbcProperties}' -chunksize ${ChunkSize} -import

'${ImportName}' -execute -output '${ErrorDump}' "
```

Windows:

```
set Operation=-service "https://api.anaplan.com" -auth
```

JDBC

```
"https://us1a.app.anaplan.com"
-debug -service %ServiceUrl% -auth %AuthUrl% -workspace %WorkspaceId% -model %ModelId% -file %FileName%
-jdbcproperties %/path/to/JdbcProperties% -import %ImportName% -execute -output %ErrorDump%
```

The *JdbcProperties* file contains the connection details including the path to the database, username, password, and the query string.

Note: A properties file can contain only one query. If you have multiple queries, create a properties file for each query. Do not include any comments in the property file, that will cause an error.

Example Properties File

```
# JDBC Connection string (Oracle, Mysql, H2, etc.)
jdbc.connect.url= "jdbc:mysql://localhost:3306/apcustomer”

# JDBC login username
jdbc.username=user1

#JBC login password
```

```
jdbc.password=dbpasswordforuser1

# (Optional) JDBC Fetch size, might not be applicable for all JDBC flavors.

jdbc.fetch.size=10

# Boolean flag, needs to be true if provided {jdbc.query}
is a # stored-procedure/function call.
# Defaults to false if no value provided, or invalid.

jdbc.isStoredProcedure=false

# SQL query could be Select or a Stored-Procedure/Function call. In case
of latter, # require {jdbc.isStoredProcedure} above set to "true". # NOTE:
SQL query needs to be parametrized with ? placeholders and values for #
placeholders needs to be provided below in {jdbc.params}.

jdbc.query=” SELECT * FROM Anaplan where col1 = ? and col2 = ?” #

JDBC parameters for parametrized SQL query in {jdbc.query} above.

# NOTE: Number of parameters must equal to the number of parameters provided in {jdbc.query} above.

jdbc.params= 1234,ABC
```

Where:

jdbc.connect.url	This is the location of your database instance. This location does not have to be on the local host if the machine connecting to the database has access to the network location. To connect to an ODBC source (no longer supported in JRE 8+), you must configure your URL as follows: "*jdbc:odbc:location_of_instance*"
jdbc.username	The database username. If no user name or password is required by the source, omit the *jdbcuser* parameter.
jdbc.password	The database password. If no username or password is required by the source, omit the jdbc password parameter.
jdbc.fetch.size	Limits the number of rows that will be returned by the query.

jdbc.isStoredProcedure	Boolean value that indicates whether the following statement (jdbc.query) is a stored procedure or function call.
jdbc.query	Specifies the query type jdbc.query = "SELECT * FROM Anaplan where col1 = ? and col2 = ?" is the query to run.
jdbc.params	JDBC parameters for parametrized SQL query in {jdbc.query} above. jdbc.params = 1234,ABC

Export to a database using JDBC

Using the appropriate JDBC driver, you can export directly to a compatible database.

Warning: When you export to a database, this is a potentially destructive action. You should only use this feature if you have a clear understanding of database operation.

Warning: The Anaplan Connect JDBC export feature only supports the SQL commands below:

- INSERT IGNORE - Insert into a database without creating duplicate values. l REPLACE - Replace current values with the new data.
- INSERT - Insert information into the database. May generate errors if duplicate values exist. The target database determines which rows are duplicate based on user-defined primary key settings. Anaplan Connect doesn't participate in this determination.

Additional SQL commands like DELETE or stored procedures are not supported.

Creating the Anaplan Connect Script

In order to use the JDBC export feature, your Anaplan Connect script should contain your model's info, as well as the path to your properties file.

Mac:

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
export '${ExportName}' -execute jdbcproperties ${/path/to/JdbcProperties}"
```

Windows:

```
set Operation=-service "https://api.anaplan.com" -auth
"https://us1a.app.anaplan.com"
```

```
-debug -service %ServiceUrl% -auth %AuthUrl% -workspace %WorkspaceId% model %ModelId% -export
%ExportName% -execute -jdbcproperties %/path/to/JdbcProperties%
```

Properties File

The *JdbcProperties* file contains the connection details. These connection details include the path to the database, username, password, and the query string. The example below includes query string examples for all Anaplan Connect JDBC export types: REPLACE, INSERT, and INSERT IGNORE. Once the job is complete, Anaplan Connect will return a count of all successful and unsuccessful updates to the database. **Important considerations:**

- Anaplan Connect property files use commas as delimiters between property values. In order to avoid problems, Anaplan column names should not have commas. l All target DB table columns should be of **char** data type.
- A properties file can contain only one query. If you have multiple queries, create a properties file for each query.
- The target database's columns should NOT have a **SQL NOT NULL** constraint. It's possible for exported data to have no data for some columns. In this case, the Anaplan Connect export feature will send **NULL** values.
- When exporting data from an Anaplan list, the first column of the exported file may have a blank header. If your first column has a blank header, there is no need to provide the **jdbc.params** variable in the properties files. Anaplan Connect will automatically map exported columns to JDBC query arguments in sequence from left to right.

Note: JDBC write speed is dependent on the respective JDBC driver and database settings. Anaplan Connect does not control or moderate this speed.

Example Properties File

```
# JDBC Connection string (Oracle, Mysql, H2, etc.)
jdbc.connect.url=jdbc:mysql://localhost:8080/testdatabase

# JDBC login username
jdbc.username=user1
```

```
#JDBC login password
jdbc.password=dbpasswordforuser1

# Use INSERT queries to export your data into database
# Use the INSERT IGNORE command rather than the INSERT command. If a record doesn't duplicate an existing record, then SQL inserts it as usual. If the record is a duplicate, then the IGNORE keyword tells SQL to discard it silently without generating an error.
# Use the REPLACE command rather than the INSERT command. If the record is new, it is inserted just as with INSERT. If it is a duplicate, the new record replaces the old one.
# SQL query example jdbc.query=INSERT IGNORE INTO `table_name` (`column1`,`column2`) Values (?,?); or jdbc.query=REPLACE into `table_name` (`column1`,`column2`) Values (?,?); or jdbc.query=INSERT into `table_name` (`column1`,`column2`) Values (?,?);

# -----NEW PARAMETER-----
# JDBC parameters for parametrized SQL write in {jdbc.query} above.
# NOTE: The number of parameters must be equal to the number of parameters provided in the {jdbc.query} above.
#     The parameter names must match column names from Anaplan export action.
# The JDBC parameters below will be mapped starting from left to right to the '?' in {jdbc.query} above.
#       In this example, the exported data has columns names column1 and column2. These will be mapped
#       starting from left to right to each of the '?' above.
jdbc.params=column1,
column2
```

Where:

jdbc.connect.url	This is the location of your database instance. This location does not have to be on the local host if the machine connecting to the database has access to the network location.
jdbc.username	The database username. If no user name or password is required by the source, omit the *jdbcuser* parameter.
jdbc.password	The database password. If no user name or password is required by the source, omit the *jdbcpassword* parameter.
jdbc.query	Specifies the query type. This may either be a REPLACE, INSERT IGNORE, or INSERT operation. jdbc.query = "REPLACE INTO `export_large_exports_1` (`column1`,`column10`) Values (?,?);" is the query to run.
jdbc.params	JDBC parameters for parametrized SQL query in {jdbc.query} above. jdbc.params = column1,column2

Loadclass Parameter

Anaplan Connect 1.4.4 has been updated to automatically use the JDBC driver. There is no need to provide loadclass parameter in your scripts. If your old scripts still contain a loadclass parameter, they will continue to work, but you will see a warning message. We recommend that you remove loadclass parameters nonetheless. In an upcoming version of Anaplan Connect, loadclass parameter will be removed entirely and scripts containing loadclass parameter will fail.

Database Driver Installation

If you are using MySQL database:

- You do not have to install the database driver in the /lib directory for any of your Anaplan Connect data loads.

For all other databases:

- You should install the appropriate driver in the /lib directory. If the appropriate driver is not installed in the /lib directory, you will get an error message to install it.
- There is a known bug for *Exports*. If the appropriate driver is not installed in the /lib directory, you will receive an invalid error message and your scripts will fail.

Create a Script to Run Other Actions

Example Batch file for Delete

Basic Authentication

```
@echo off rem This example deletes obsolete
customers from a list set
AnaplanUser=firstname.lastname@company.com set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-service
"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-action "Delete from Customers Using Obsolete" -execute rem *** End of settings - Do not edit
below this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 cd %~dp0 if
not %AnaplanUser% == "" set Credentials=-user %AnaplanUser% set
Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model %ModelId%
%Operation% @echo %Command% cmd /c %Command% pause
```

Certificate Authentication

```
@echo off rem This example deletes obsolete customers from a list set
CertPath="C:\certs\cert.pem" set KeyStorePath="C:\certs\AC1.4_keystore.jks"
set KeyStorePass="keystorepass" set KeyStoreAlias="keystorealias" set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-service

"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-action "Delete from Customers Using Obsolete" -execute rem *** End of settings - Do not edit below
this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 cd %~dp0 set Credentials=-
certificate %CertPath% -keystore %KeyStorePath% -keystorepass %KeyStorePass% keystorealias
%KeyStoreAlias% set Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model

%ModelId% %Operation% @echo %Command% cmd /c %Command% pause
```

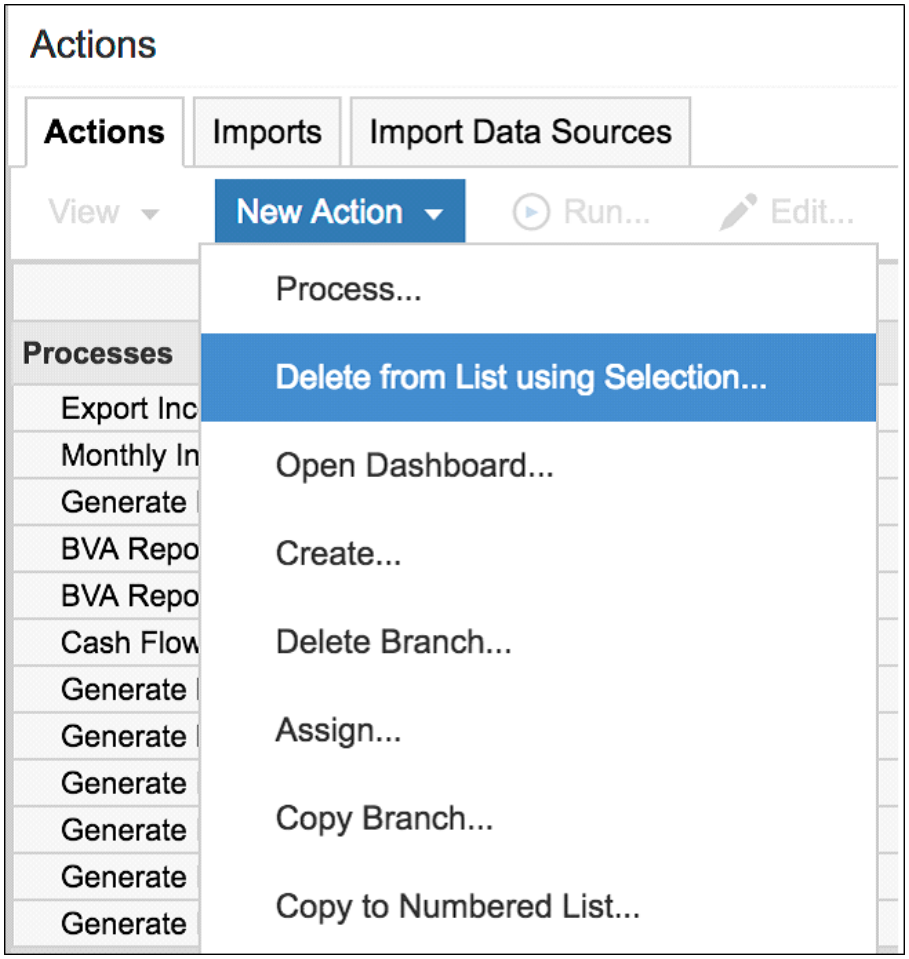
Set the Delete operation to delete items from a list

This example shows how to automate deleting items from a list based on Boolean criteria.
For example, you can automatically delete items in a list that are now obsolete, such as customers with a rating less than, or equal to, 2. The Obsolete line item has Boolean data type and must be set up to be a formula:

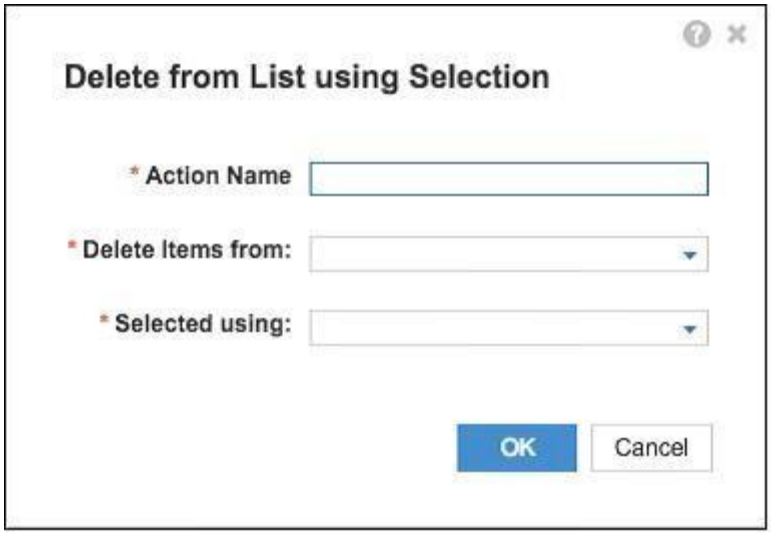
```
Obsolete = Rating <= 2
```

Line items of Boolean data type that only have the dimensionality of the list can be used as the criteria to determine which items to delete.

1. In Anaplan, go to **Model Settings > Actions**.
2. On the **New Action** list, click **Delete from List using Selection**.



The Delete from list using Selection dialog appears.



3. In the **Action Name** box, type the text that you want to appear on the button.
4. On the **Delete Items from** list, select the list (for example *Customers*) from which you are deleting items.
5. On the **Selected using** list, select the line item that contains the selection criteria (for example *Customer Rating.Obsolete*).
6. Click **OK**. The action appears under **Other Actions**. Use the name of this action in your batch script. See Delete from List using Selection in Anapedia. Example:

```
set Operation=-service "https://api.anaplan.com" -auth
"https://us1a.app.anaplan.com"

-action "Delete from Customers Using Obsolete" -execute
```

where:

-action "Delete from Customers Using Obsolete"	an Other Action (neither Import nor Export) named Delete from Customers Using Obsolete
-execute	runs the Delete

For Linux and Mac OS examples of delete scripts, see Appendix G.

Create a Script to run a Process

A Process is a combination of Imports, Exports, and/or Deletes.

The process must exist within Anaplan before calling it in the Anaplan Connect script.

4. An example batch file for a process that contains Import and Export actions shown below.

Note: A process can contain delete actions.

Basic Authentication

```
@echo off rem This example runs a Process that contains multiple Export
actions set AnaplanUser="firstname.lastname@company.com" set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-service

"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-file "file1.txt" -put"C:\Files\ImportModule.csv" -file "file2.txt" -put "C:\Files\example.csv" process
"myprocess" -execute -file "ExportList" -get "C:\Files\ExportList.csv" -file "ExportModule"
-get "C:\Files\ExportModule.csv" -output "C:\MyDirectoryForImportDumpFiles" rem *** End of
settings -Do not edit below this line *** setlocal enableextensions enabledelayedexpansion
|| exit /b 1 cd %~dp0 if not %AnaplanUser% == "" set Credentials=-user %AnaplanUser% set
Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -model %ModelId%
%Operation% @echo %Command% cmd /c %Command% pause
```

Certificate Authentication

```
@echo off rem This example runs a Process that contains multiple
Export actions set CertPath="C:\certs\cert.pem" set
KeyStorePath="C:\certs\AC1.4_keystore.jks" set
KeyStorePass="keystorepass" set KeyStoreAlias="keystorealias" set
WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-service
"https://api.anaplan.com" -auth "https://auth.anaplan.com"
-file "file1.txt" -put"C:\Files\ImportModule.csv" -file "file2.txt" -put "C:\Files\example.csv" process
"myprocess" -execute -file "ExportList" -get "C:\Files\ExportList.csv" -file "ExportModule"
-get "C:\Files\ExportModule.csv" -output "C:\MyDirectoryForImportDumpFiles" rem *** End of settings -
Do not edit below this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 cd
%~dp0 set Credentials=-certificate %CertPath% -keystore %KeyStorePath% -keystorepass %KeyStorePass%
keystorealias
%KeyStoreAlias% set Command=.\AnaplanClient.bat %Credentials% -workspace %WorkspaceId% -
model %ModelId% %Operation% @echo %Command% cmd /c %Command% pause
```

Set the Process Operation

```
set Operation=-service "https://api.anaplan.com" -auth "https://us1a.app.anaplan.com"
-file "file1.csv" -put "C:\Files\ImportModule.csv" -file "file2.csv" -put "C:\Files\example.csv" -
process "myprocess" -execute -file "ExportList" -get "C:\Files\ExportList.csv"
-file "ExportModule" -get "C:\Files\ExportModule.csv" -output "C:\MyDirectoryForImportDumpFiles"
```

Where:

-file "file1.csv" -put "C:\Files\ImportModule.csv"	Uploads the local file *C:\Files\ImportModule.csv* to the Anaplan server so that its data can be stored into the Import data source named *file1.csv*.
-file "file2.csv" -put "C:\Files\ImportModule.csv"	Uploads the local file *C:\Files\ImportModule.csv* to the Anaplan server so that its data can be stored into the Import data source named *file2.csv*.
-process "myprocess" -execute	Runs the process named *myprocess*
-file "ExportList" -get "C:\Files\ExportList.csv"	Downloads the data last exported by the export *ExportList* to the local file *C:\Files\ExportList.csv*
-file "ExportModule" -get "C:\Files\ExportModule.csv"	Downloads the data last exported by the export *ExportModule* to the local file *export/ProcessExportModule.csv*
-output "C:\MyDirectoryForImportDumpFiles"	Generates a file for each import action within the specified directory that lists one or more import errors

For Linux and Mac OS examples of process scripts, review Appendix G.

End Users versus Workspace Administrators

An end user can run the same actions through Anaplan Connect that the end user can run manually.

	End user can run..	Workspace Admin can create and run..
Import	• model-to-model import l list imports - requires write access to the target list • module imports requires write access to the target module	any type of import, including imports that involve uploading external files or data
Export	Requires read access to the module or list.	export
Delete	Requires write access to the list.	delete
Process	Requires access to the actions in the process	process
Info		• Can change the **model**, unless the role is *No Access* to a particular model, which also prevents the Workspace Admin from finding the model. • Can have a role that has access to no **module** but can grant self rights to the module.

Scheduling an import or export

A batch file that runs the import or export can be scheduled to run at a specific time, as a one-time operation, or recurring at the interval you choose, such as daily, weekly, or monthly. The scheduler is not part of Anaplan Connect, and the scheduling program and set-up depends on your operating system; the computer must be running at the scheduled time.
In the batch file, the password needs to be appended to the Anaplan user name and enclosed in double quotes. Alternatively use certificate authentication.

Windows

```
set AnaplanUser=firstname.lastname@company.com:"mysecretpassword"
```

Optionally, you can remove the pause command at the end of the batch file.

The pause command leaves the messages on the screen that record what the batch file has done.

Scheduler for Windows XP

This example shows the steps involved on a Windows XP operating system to schedule an import on a specific day and time, once a month:

l Start > All Programs > Accessories > System Tools > Scheduled Tasks > Add Scheduled Tasks > Next > Browse

```
C:\anaplan-connect-1-4\nameofscript.bat
```

- Monthly > Next > The First Monday at 05.00 > Enter name & password for the PC > Finish

Scheduler for Windows 7

The Scheduler in Windows 7 is almost the same as XP:

l Start > All Programs > Accessories > System Tools > Task Scheduler > Create Basic Task > *Name the task* > Next > *Set when to trigger the task* > Next > Select Start a program and browse to *C:\anaplanconnec\nameofscript.batt* > Next > **Finish**

Linux or Mac OS

Consider using a job scheduling utility for UNIX-like operating systems, such as cron.

Use Anaplan transactional API capabilities to access model data and metadata

With the added features below, you can use transactional APIs to read, display, and update model data without import or export actions. You can:

- Get a list of workspaces, along with space allocated and space used.
- Display a list of models and space used.
- Display a list of modules and views in your Anaplan model.
- Get data from a module view without export actions (up to 1M cells)
- Display items from model lists without export actions (up to 1M list items).

Display lists of workspaces, space allocated, and space used

The commands below return list of Anaplan workspaces accessible to you, the space (bytes) allocated, and the space (bytes) taken by each workspace.

Example script:

```
set AnaplanUser="username:password" set ServiceUrl="https://api.anaplan.com" set AuthUrl="https://us1a.app.anaplan.com" set Operation=-debug -service %ServiceUrl% -auth %AuthUrl% -W execute rem *** End of settings - Do not edit below this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Example scripts for Linux or Mac OS:

See "Appendix G: Linux and MacOS Scripts" on page 67.

Display lists of models and space used

The commands below return a list of Anaplan models accessible to you, and the space (bytes) taken by each model.

Example script:

```
set AnaplanUser="username:password" set ServiceUrl="https://api.anaplan.com" set
```

```
AuthUrl="https://us1a.app.anaplan.com" set Operation=-debug -service %ServiceUrl% -auth %AuthUrl%
models rem *** End of settings - Do not edit below this line *** setlocal enableextensions
enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat
%Credentials% %Operation%
@echo %Command% cmd /c %Command% pause
```

Get a complete list of modules and saved views within your model

The commands below return names and internal IDs of all modules and views accessible within the specified Anaplan model.

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set WorkspaceName="workspacename" set
ModelName="CB0A5A4D5C5943B5837FF42C5FAA95E1" set Operation=-debug -service %ServiceUrl% -auth %AuthUrl%
-execute -workspace
%WorkspaceName% -model %ModelName% -views -execute rem *** End of settings - Do not edit below this line
*** setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser% set
Command=AnaplanClient.bat %Credentials% %Operation%
@echo %Command% cmd /c %Command% pause
```

Display data from saved views without using export actions (up to one million cells)

The commands below return data from a specified module view without an export action. Cells display in a format similar to 'Tabular Single Column' from an export action. The transactional API here reeads up to 1M cells.

If your module view has more than 1M visible cells, Anaplan Connect will display an error message.

Example script (single column format):

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set WorkspaceName="WorkspaceName" set ModelName="ModelName"
set FilePath="localfilesystempath" set ModuleName="modulenameorid" set ViewId="viewnameorid" set
Operation=-debug -service %ServiceUrl% -auth %AuthUrl% -workspace %WorkspaceName% -model %ModelName%
-module %ModuleName% -view %ViewId% execute -get:csv_sc %FilePath% setlocal enableextensions
enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat
%Credentials% %Operation%
@echo %Command% cmd /c
%Command% pause
```

Example script (multiple column format):

Use the commands below to return data from a specified module view without an export action. Cells display in a format similar to 'Tabular Multi Column' from an export action. The transactional API here reeads up to 1M cells. If your module view has more than 1M visible cells, Anaplan Connect will display an error message.

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set WorkspaceId="8a1234567897c12b014bf01234567890" set
ModelId="CB0A5A4D5C5943B5837FF42C5FAA95E1" set FilePath="localfilesystempath" set
ModuleName="modulenameorid" set ViewId="viewnameorid" set Operation=-debug -service %ServiceUrl% auth
%AuthUrl% -workspace
%WorkspaceId% -model %ModelId% -module %ModuleId% -execute -get:csv_mc
%FilePath% setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user
%AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Display data from saved views with parameterization

You can apply parameters to the cell data read operation. To do this, specify members or items for dimensions within the page axis of the saved view. For example, if you have a saved view with “Product” in the page axis, to display its data, 'Product=Product5', use the ‘Pages’ parameter.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set WorkspaceId="Workspace name" set ModelId="Model name" set
FilePath="localfilesystempath" set ModuleName="modulenameorid" set ViewName="default" set
Pages="pageDimensionName:dimensionMemberName" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace %WorkspaceId% -model %ModelId% -mo %ModuleName% -view %ViewName% -pages
%Pages% -execute -get:json %FilePath% rem *** End of settings - Do not edit below
this line *** setlocal enableextensions enabledelayedexpansion || exit /b 1 set
Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat %Credentials%
%Operation%
@echo %Command% cmd /c
%Command% pause
```

Display properties and metrics for model lists

Use the commands below to return detailed descriptions or metadata for specified model list.

Note: this command only returns metadata, and not the actual items from the list.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="Workspace name" set ModelId="Model name" set
FilePath="localfilesystempath" set ListId="listid" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace %WorkspaceId% -model %ModelId% -l %ListId% -get:csv %FilePath% rem *** End of
settings - Do not edit below this line *** setlocal enableextensions enabledelayedexpansion || exit /b
1 set Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Display items from your model lists without using export actions (up to one million items)

Use the commands below to return data (items) from a specified Anaplan list without an export action. This transactional API supports up to 1M items. If your Anaplan list has more than one million items, Anaplan Connect will display an error message.

Note: Use the “-execute:all” parameter to read custom properties from lists. To read list items IDs, name, code and the parent properties, use the “-execute” parameter.

Example script (with include all):

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="listid" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace
%WorkspaceId% -model %ModelId% -l %ListId% -execute:all -get:csv
%FilePath% setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user
%AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%
@echo %Command% cmd /c %Command% pause
```

Example script (without include all):

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="listid" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace %WorkspaceId% -model %ModelId% -l %ListId% -execute -get:csv %FilePath%
setlocal enableextensions  enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser%
set Command=AnaplanClient.bat %Credentials% %Operation%
@echo %Command% cmd /c
%Command% pause
```

Add new items to Model Lists

Use this command to add new items to a model list without requiring an intermediate import action. The supported data sources are CSV files, JSON files and JDBC sources.

Note: This operation uses the transactional list item API. For a large volume of list items, Anaplan Connect batches these items into multiple API calls. There is a small chance that another process may update list data in between such API calls.

- If the source has columns with different headings, then the parameter -itemmappingproperty can be used to map the data to the list properties.
- If the optional parameter -item mappingproperty is not provided, the data source must have columns exactly matching the list custom property names ('Name', 'Code', other customer properties).
- The column headings are not case sensitive. The data source column headings must be one of following types, otherwise the column will be ignored and will not be mapped: ○ Name ○ Code ○ Parent ○ List property ○ Subset of a list

Note: The optional -output parameter can be used to write error records to disk in either CSV or JSON format. This operation adds new list items. If any item in the source is already present in the List, it will be flagged as an error. To add and update list items in a single command, use the Upsert operation.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="44isted" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace
%WorkspaceId% -model %ModelId% -list %<list_id_or_name>% -execute -putItems(csv|json|jdcb)
<path_to_local_file>
```



```
%FilePath% setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user
%AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Update existing items in Model Lists

Use this command to update list items without an import action. The supported data sources are CSV files, JSON files and JDBC sources.

Note: This operation uses the transactional list item API. For a large volume of list items, Anaplan Connect batches these items into multiple API calls. There is a small chance that another process may update list data in between such API calls.

- If the source has columns with different headings, then the parameter `-itemmappingproperty` can be used to map the data to the list properties.
- If the optional parameter `-item mappingproperty` is not provided, the data source must have columns exactly matching the list custom property names ('Name', 'Code', other customer properties).
- The column headings are not case sensitive. The data source column headings must be one of following types otherwise the column will be ignored and will not be mapped:
  - Name
  - Code
  - Parent
  - List property
  - Subset of a list

The optional -output parameter can be used to write error records to disk in either CSV or JSON format.

Note: This operation updates existing list items. If any item in the source is not already present in the list, it will be flagged as an error. To add and update list items with a single Anaplan Connect command, use the upsert operation.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="46isted" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace
%WorkspaceId% -model %ModelId% -list %<list_id_or_name>%  <list_id_or_name> -execute
upsertItems⊗csv|json|jdcb) <path_to_local_file>
%FilePath% setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user
%AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Upsert (add and update combined) items to Model Lists

Use this command to add as well as update list items without an import action. The supported data sources are CSV files, JSON files and JDBC sources.

Note: This operation uses a combination of transactional list item 'add' and 'update' APIs. For a large volume of list items, Anaplan Connect batches these items into multiple API calls. There is a small chance that another process may update list data in between such API calls.

- If the source has columns with different headings, then the parameter `-itemmappingproperty` can be used to map the data to the list properties.
- If the optional parameter `-item mappingproperty` is not provided, the data source must have columns exactly matching the list custom property names ('Name', 'Code', other customer properties).
- The column headings are not case sensitive. The data source column headings must be one of following types otherwise the column will be ignored and will not be mapped: ○ Name ○ Code ○ Parent ○ List property ○ Subset of a list

Note: The optional -output parameter can be used to write error records to disk in either CSV or JSON format.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="listid" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace %WorkspaceId% -model %ModelId% -list %<list_id_or_name>%  <list_id_or_name>
execute -upsertItems:(csv|json|jdcb) <path_to_local_file> %FilePath% setlocal enableextensions
enabledelayedexpansion || exit /b 1 set Credentials=-user %AnaplanUser% set Command=AnaplanClient.bat
%Credentials% %Operation%

@echo %Command% cmd /c %Command% pause
```

Delete items from model lists

Use this command to delete list items by providing ID or Code for the item, without a saved delete action in your model.

Notes:

(1) The model list item 'Name' is not supported. List items can be specified from CSV files, JSON files and JDBC sources.

(2) This operation uses the transactional list item 'delete' API. For a large volume of list items, Anaplan Connect batches these items into multiple API calls. There is a small chance that another process may update list data in between such API calls.

- If the source has columns with different headings, then the parameter `-itemmappingproperty` can be used to map the data to the list properties.
- If the optional parameter `-item mappingproperty` is not provided, the data source must have the columns names 'Id' and 'Code'.
- The column headings are not case sensitive.
- The optional -output parameter can be used to write error records to disk in either CSV or JSON format.

Example script:

```
set AnaplanUser=username:password set ServiceUrl="https://api.anaplan.com" set
AuthUrl="https://us1a.app.anaplan.com" set Workspace="WorkspaceName" set Model="ModelName" set
FilePath="localfilesystempath" set ListId="listid" set Operation=-debug -service %ServiceUrl% -auth
%AuthUrl% -workspace
%WorkspaceId% -model %ModelId% -list %<list_id_or_name>%  <list_id_or_name> -execute
deleteItems:(csv|json|jdcb) <path_to_local_file>
%FilePath% setlocal enableextensions enabledelayedexpansion || exit /b 1 set Credentials=-user
%AnaplanUser% set Command=AnaplanClient.bat %Credentials% %Operation% @echo
%Command% cmd /c %Command% pause
```

Log Files in Anaplan Connect

Logging files generated by Anaplan Connect in v1.4 and later use logback to provide file output.

The log line format is:

YYYY-MM-DD hh:mm:ss <Class Path> <PID> |-- <Logging statement description> Where:

Element	Description
YYYY	Year in four digit format.
MM	Month in two digit format
DD	Day of the month in two digit format
hh	Hour in 24-hour format
mm	Minutes
ss	Seconds
<Class Path>	The shortened class path. For example: DEBUG [c.a.client.Program :1594] or INFO [a.BasicAuthenticator:26]
<PID>	The UNIX process ID. Note: On Windows systems, this entry may be blank.

|--	A delimiter between the basic logging information and the logging statement description.
<Logging statement description>	A descriptive logging statement.

Example:

```
2018-05-06 20:30:18 DEBUG [c.a.client.Program :1595 ] 96461 |--             Anaplan
Connect 2.0.0-release
2018-05-06 20:30:18 DEBUG [c.a.client.Program :1596 ] 96461 |--            Java
HotSpot(TM) 64-Bit Server VM (Oracle Corporation)/ (25.60-b23)/
2018-05-06 20:30:18 DEBUG [c.a.client.Program :1598 ] 96461 |--            (Mac OS)
Xx86_64)/10.13.4
2018-05-06 20:30:18 INFO [c.a.client.Service :88 ] 96461 |-Initializing
Service...
2018-05-06 20:30:18 INFO [a.BasicAuthenticator:26     ] 96461 |-Authenticating
via Basic...
2018-05-06 20:30:21 INFO [c.a.c.ServerFile      :259 ] 96461 |-- Uploading file:
/Users/USERX/Downloads/gross_sales_actuals.csv
2018-05-06 20:30:33 DEBUG [c.a.c.ServerFile           :306 ] 96461 |--
Uploaded chunk: 0 (size=30MB)


2018-05-06 20:31:59 INFO [c.a.client.Program :464 ] 96461 |-file
"/Users/USERX/Downloads/gross_sales_actuals.csv" has been uploaded as
gross_sales_actuals_DEC15.csv.
2018-05-06 20:32:00 INFO [c.a.c.TaskFactory           :116 ] 96461 |--
Creating Import task: 112000000030
```

```
2018-05-06 20:32:02 INFO [c.a.client.Task :207 ] 96461 |-status: Retrieving
data from gross_sales_actuals_DEC15.csv

2018-05-06 20:32:36 INFO [c.a.client.Task :207 ] 96461 |-status: Generating
failure dump (100.0%)

2018-05-06 20:32:46 INFO [c.a.client.Task :207 ] 96461 |-status: Complete.
(100.0%)


2018-05-06 20:32:46 INFO [c.a.client.Task :230 ] 96461 |-operation was
successful.
2018-05-06 20:32:46 INFO [c.a.client.Task :3880 ] 96461 |-list_USERX: 2

(0/2) rows successful, 1179070 ignored


2018-05-06 20:32:46 INFO [c.a.c.ServerFile            :109 ] 96461 |--

Downloading file /Users/USERX/projects/anaplan-connect/errors.txt
2018-05-06 20:32:46 DEBUG [c.a.c.TaskResult           :127 ] 96461 |--

Fetching Import action's dump file chunks for
   task=A96ADE416B594080BD8E7F7FB25BA51B

2018-05-06 20:32:47 DEBUG [c.a.c.TaskResult           :138 ] 96461 |--
Downloading dump data-chunk 0
2018-05-06 20:32:57 INFO [c.a.client.Program :714 ] 96461 |-- Dump file written

to "errors.txt"
```

Retries in Anaplan Connect

Anaplan Connect automatically retries API and JDBC (export) calls for:

- Network connection and Input/Output (I/O) exceptions. If you lose connectivity while running Anaplan Connect (typically due to a network error or socket time-out), you will receive an I/O error.
- 5XX errors returned by Anaplan APIs: Anaplan APIs can return 5xx error in case something unusual happens.
- JDBC exceptions: when a database is unavailable upon Anaplan Connect script's initial run.

Refer to Appendix C, Optional Parameters for more detailed information on retry parameters.

Troubleshooting Tips

Debug Information

Symptom	Remedy
Error message: “Array index out of bounds: 0”	Make sure the import file has only one type of column separator as Anaplan Connect supports only one when you upload a file to Anaplan.

Error message: *".\AnaplanClient.bat is not recognized as an internal or external command, operable program or batch file."*	Move the .bat file to the root of your Anaplan Connect installation. This location guarantees that Anaplan Connect can use any batch file you create.

To get verbose command-line output that might be useful for debugging, include the **-debug** argument at the beginning of the Operation statement. If your script fails to run, enable debug logging and save the output to a text file. If you raise a Support case for assistance, providing the output file of your debug logging will aid the Support team in helping to resolve the problem.

```
set Operation=-service "https://api.anaplan.com" -auth
"https://us1a.app.anaplan.com" "-debug -file 'file-to-import.csv' -put
'/path/to/anaplan-connect/file-to-import.csv'
-import 'Organization from Salesforce' -execute -output 'MyImportErrors.txt'"
```

Symptoms and Remedies

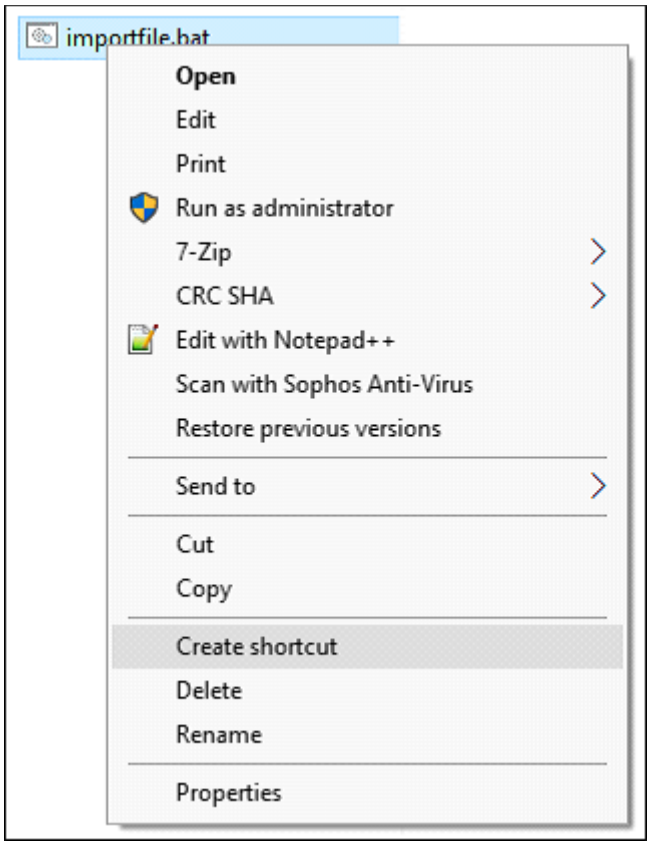
Symptom	Remedy
	filename.sh
Model-to-model import not running or not recognized.	Make sure there is a space before and after the / in the syntax import "*Target Module from Source Model / Source Module*"
If you're unable to perform a Delete action on items from a list	Check that the list does not contain summary items or subtotals. Such lists cannot use the Delete action (due to the difficulty of dealing with *orphaned* subtotals that lack children). Lists that have parent hierarchies or top-level items can use the bulk delete action, provided that the list that you are editing does not have subtotals.
If you're performing a SQL query from a Windows machine with the -jdbc.query option, and using the percent character (%) as the wildcard character in a pattern for the like operator	The Windows command processor might perform variable substitution on an expression like '%a%', even though no variable a has been defined, resulting in an empty pattern. If so, escape the % sign with %%. For example, %a% is escaped by %%a%%

You get a retry error.	The retrytimeout parameter has a maximum value of 60 seconds.

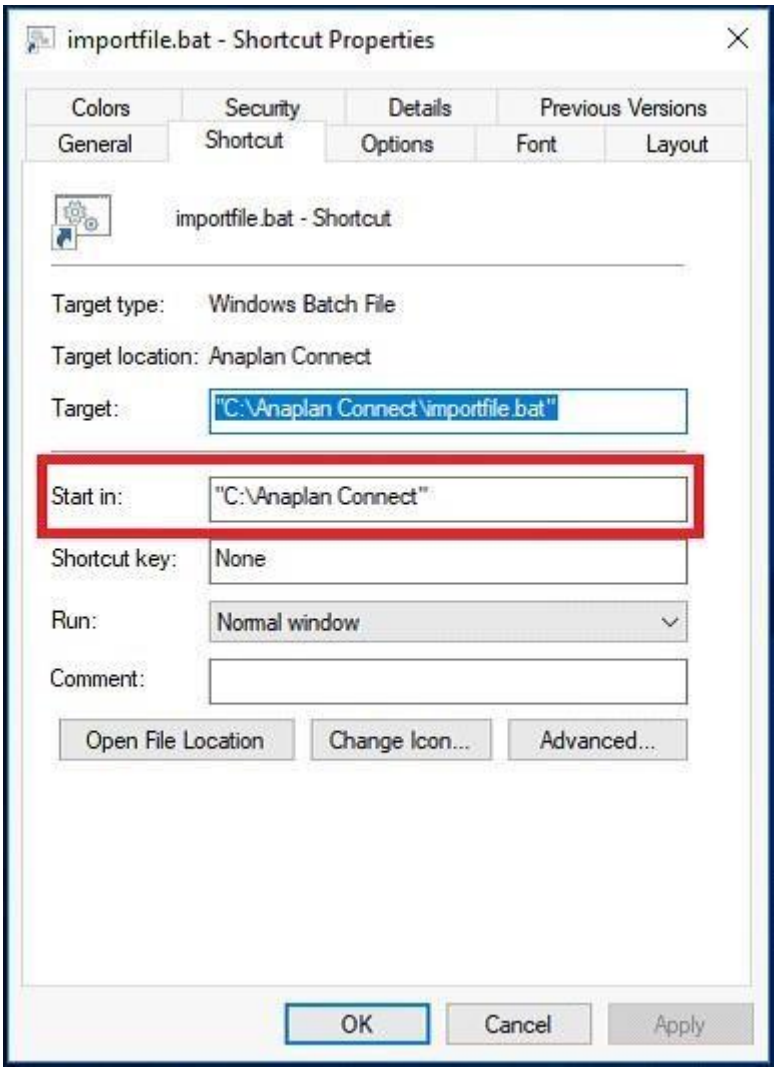
Appendix **A: Network Drive as Location for Anaplan Connect**

This example is for Windows.

1. Put a copy of example.bat (in the Anaplan Connect examples folder) into the main Anaplan Connect folder.
2. Replace line 13: *cd %~dp0* with: *pushd %~dp0*
3. Before the pause line, insert **popd**.
4. Create a shortcut to the batch file in the same directory. Right-click and select **Create shortcut**.



5. Once the shortcut is created, right-click on the shortcut and select **Properties**.
6. On the **Shortcut** tab, in the **Start** in box type the local directory that runs Anaplan Connect. For example, *%USERPROFILE%* can substitute the user's profile folder.



When given a UNC path (*computer**share*\...), the **pushd** maps the share to a drive, typically Z: or the last unmapped drive letter. The **popd**unmaps the drive and returns to the original location. If the command window is closed before the program completes, the drive remains mapped.

Appendix **B: Java Compatibility**

Anaplan Connect supports Java 8, 11 and 17. Note that we don't support Java 6 and 7.

We strongly recommend you upgrade to Java 8 to benefit from the security offered by TLS 1.2 if your organization uses Java 6.x or 7.x with Anaplan Connect.

ODBC is officially deprecated in Java 8. Make sure you update any ODBC connections to use JDBC. See Appendix D: JDBC for Oracle, Access, and Excel and Appendix E: Import a JDBC Connection for a Microsoft SQL Server Database.

Use one of the options below to upgrade to Java 8.

Create a shell script to set the JAVA_HOME environment variable

Write a shell script that sets the **JAVA_HOME** environment variable to the location of the Java 8 Runtime Environment you want to use. You only need to carry out this change to run the Anaplan Connect script.

Create a replacement script

If you've already installed Anaplan Connect you can create your own replacement script. The instructions in this section do not change the version of Java you use for your other applications.

To create the replacement script:

1. Navigate to the Anaplan Connect directory. For example, on Windows, the directory might be *C:\anaplanconnect-1-4*.
2. Make a backup copy of the script that calls Anaplan Connect:
   - **Windows**: Make a copy of *AnaplanClient.bat* and name it *AnaplanClient.bat-OLD*. | **Linux/MacOS**: Make a copy of *AnaplanClient.sh* and name it *AnaplanClient.sh-OLD*.
3. Edit the script that calls Anaplan Connect:
   - **Windows**: In **AnaplanClient.bat** replace *%JAVA%* with the version directory of Java 8 to use for Anaplan Connect.
   - **Linux/MacOS**: In **AnaplanClient.sh** replace *${java}* with the version directory of Java 8 to use for Anaplan Connect.

	Windows.bat file	Linux/MacOS.sh file
Original	rem Start the Java virtual machine "%JAVA%" %JAVA_OPTS% -classpath "%CP%" com.anaplan.client.Program %*	# Start the Java virtual machine exec ${JAVA_OPTS} -classpath "${classpath}" "${java}" com.anaplan.client.Program "$@"
Change to Java 8	rem Start the Java virtual machine "C:\Program Files\Java\jre1.8.0_ 66\bin\java" %JAVA_ OPTS% -classpath "%CP%" com.anaplan.client.Program %*	# Start the Java virtual machine for MacOS involves /Library exec "/Library/Java/JavaVirtualMachines/jdk1.8.0_60. jdk ${JAVA_OPTS} -classpath/Contents/Home/bin/java" "${classpath}" com.anaplan.client.Program "$@"

The directory name on your computer might differ from these examples. For more information, visit the Data Integration Knowledge Base on Anaplan Community.

Appendix **C: List of all Operation Commands**

Navigate to the installation directory and type the following to get a list of the operation commands:

Windows	C:\Windows\AnaplanConnect\AnaplanClient.bat -version Java AnaplanClient -version
Java	AnaplanClient -version
Linux/Mac OS	./AnaplanClient.sh ./AnaplanClient.sh -version

The following table shows the commands for the operation line of the batch file. The abbreviated syntax can be used to reduce typing, for example -x instead of -execute. Some operations are followed by a variable, such as a path to a file. For example, -put "*C:\testdata\Europe P&L.txt" or –p "C:\testdata\Europe P&L.txt*"

Authentication Operators

Use these operators based on your authentication method: basic authentication or CA authentication.

Basic Authentication

Syntax	**Abbreviated Syntax**	**Followed by a variable, if applicable**	**What it does**
-user	-u	username:"password"	**Required**. Anaplan user name and optional password in the format username:"password". If the batch file (or shell script) does not set a value for the AnaplanUser, the program prompts the user to supply the username, then the password, enclosed in double-quotes.

CA Authentication

Option 1: Authentication using a Private Key

When using Certificate Authority (CA) authentication with your private key, the required operators are *certificate* and *-privatekey*.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-certificate	-c	Pathname on local machine	**Required**. Path to user certificate used for authentication (an alternative to using a key store)
-privatekey	-pkey	<privatekey path>:<passphrase>	**Required**. Path to user private key and passphrase used for authentication (an alternative to using a key store)

Option 2: Authentication using a Java KeyStore

When using Certificate Authority (CA) authentication with a Java KeyStore, the required operators are *-certificate* or a combination of *-keystore*, *-keystorealias*, and *-keystorepass*.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-certificate	-c	Pathname on local machine	**Required**. Path to user certificate used for authentication (an alternative to using a key store)
-keystore	-k	Pathname on local machine	**Required**. Path to local key store containing user certificate(s) for authentication

-keystorealias	-ka	Alias	**Required**. Alias of the public certificate in the specified key store
-keystorepass	-kp	Password	**Required**. Password for the key store. If this option is not provided, and the file ~/.anaplan/api-client/keystore-access.txt exists (where ~ is the user's home directory), the password is read and decoded from the contents of this file. Otherwise, the user is prompted for a password. - Note that obfuscation is the URL-encoded form of the result of: taking the exclusive -or of each of the characters in the password and the value 129

Proxy Operators

Use these operators for proxies.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-via	-v	Proxy URL	Use specified proxy
-viauser	-vu	username:password or domain/workstation/username:password	For basic proxy authentication, use username and password. For NTLM proxy authentication, use domain/workstation/username and password. Consider NTLM Authentication using JCIFS .

Required Operators

The batch file must have each of these operators in addition to an action operator.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does

-authServiceUrl	-auth	The URL for authentication (ex. "https://us1a.app.anaplan.com")	**Required**. Specifies the URL for the authentication service
-model	-m	Model ID or Model name	Select a model by ID number or model name
-service	-s		**Required**. API service endpoint.
-workspace	-w	Workspace ID or Workspace name	**Required**. Select a workspace by ID number or workspace name

Action Operators

You can specify operators for import, export, process, and delete actions.

Import

Use for importing your data into Anaplan.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-execute	-x		**Required**. Run the preceding -import, -export, -process, or action. Also gets list data.
-file	-f	File name on Anaplan server	**Required**. File name on Anaplan server. Select a server file by ID number or name.
-import	-i	Import name or ID	**Required**. Select an import by ID number or name.

-locale	-xl	ISO language and country code separated by underscores. For example, "en_US"	Specify the locale to use when performing the server operation, which affects the available data formats when parsing date values in imports, and the month names when using a specified timescale format. For details, see http://docs.oracle.com/javase/8/docs/api/java/util/Locale.html
-output	-o	Pathname on local machine	**Required**. Retrieve dump file for completed import.
-put	-p	Pathname on local machine	Upload the specified file.
-putc			Upload to specified server file from tab-separated standard input
-puts			Upload to specified server file from standard input

Export

Use to export data from Anaplan.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-execute	-x		**Required**. Run the preceding -import, -export, -process, or action. Also gets list data.
-export	-e	Export name or ID	**Required**. Select an export by ID number or name
-get	-t	Pathname on local machine	Download the specified file.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-gets			Write specified server file to standard output.
getc			Write tab-separated server file to standard output.

Process

Use the process action if you want to run a combination of actions, such as import, export, and delete.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-execute	-x		**Required**. Run the preceding -import, -export, -process, or action. Also gets list data
-process	-pr	Process name or ID	**Required**. Select a process by ID number or name.

Delete

Use to delete data from Anaplan.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-action	-a	Action name or ID	**Required**. Select a saved action. For example, Delete items from a list.
-execute	-x		**Required**. Run the preceding -import, -export, process, or -action.

Listing

Use these commands to list the appropriate output.

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-actions	-A		List available actions in selected model, such as delete actions. This list corresponds to the Other Actions (Settings tab, Actions) list, and does not include Processes, Imports, or Exports.
-exports	-E		List available exports in selected model.
-files	-F		List available files on the Anaplan server in the selected model.
-imports	-I		List available imports in the selected model. The output lists all import definitions that are available in a given model. The list of imports and exports is also available in Anaplan. To view, click **Actions** on the Settings tab.
-processes	-P		List available processes in selected model

Parameters for transactional cell data read

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does

-execute:all			Downloads list items along with all custom properties
-get:csv			Download data from a list in csv format
-get:csv_sc			Download data from a view in single column format (similar to tabular, single column format for an export action)
-get:csv_mc			Download data from a view in multi-column format (similar to tabular, multi-column format for an export action)
-get:json			Download data from a list or saved view in json format
-lists	-L		Display available lists in selected model
-list	-l		Select a list by ID number or name
-models	-M		List available models
Syntax	**Abbreviated Syntax**	**Followed by a variable, if applicable**	**What it does**
-modules	-MO		List available modules
-module	-mo		Select a module by ID number or name

-pages			Comma separated list of <page dimension id>:<dimension member id> The page selector values that identify the page to retrieve
-views	-V		List available views
-view	-vi	View name or ID	Select a view by ID number or name
-workspaces	-W		List available workspaces

Optional Parameters

Use these parameters to customize Anaplan Connect for your specific needs.

Syntax	**Abbreviated Syntax**	**Followed by a variable, if applicable**	**What it does**
-connectorproperty	-xc	Property identifier and value separated by colon. If value is ?, prompt user	Specify import data source connection property. For example, Salesforce credentials.

-debug	-d		Show detailed (verbose) output, which can help you debug any problems. See Getting Debug Information.
-emd			Get metadata for an export

Syntax	**Abbreviated Syntax**	**Followed by a variable, if applicable**	**What it does**
-help	-h		Display this help
-httptimeout	-ct		The socket timeout. If not specified, the default is 60 seconds. The minimum value can be 3 and the maximum value 300. Use directly after the -models or -model_id command. If you use this command in conjunction with -maxretrycount and -retrytimeout, use these commands in alphabetical order.
--forceRegister			Used to delete keystore and re-register the device (via OAuth device grant flow). Once approved, the device will be registered and a corresponding keystore file will be generated at your home location (by default). Note: “user.home” is the user home directory a system property set by Java VM.
-mappingproperty	-xm	Dimension and value separated by colon. If value is ?, prompt user	Specify prompt-at-runtime import mapping value

-models	-M		List available models
-model_id	-m_id		Call by model ID. This parameters bypasses checks for model (name and ID). It increases the integration run speed. Abbreviated way to call by model ID
-workspace_id	-w_id		Call by workspace ID. This parameters bypasses checks for workspace (name and ID). It increases the integration run speed. Abbreviated way to call by workspace ID
-maxretrycount	-mrc	Retry count	Maximum retry count for API calls. If not specified, the default is 3 retries. The minimum value can be 3 and the maximum value 15. Use directly after the -models or -model_id command. If you use this command in conjunction with -httptimeout and -retrytimeout, use these commands in alphabetical order. **Note: When using this parameter, specify it in your script before your -execute parameter. When you have a -put parameter, the maxretry count should be placed before -put.**
-oauth-client-id			Used to log in via OAuth2. It is required to provide the client ID. If the batch file (or shell script) does not set a value for the client-Id parameter, you'll be prompted to enter the client ID. The client ID type must be device grant. Device grant is a type of extension. It enables devices without browsers or limited
			input capability to exchange a prior device code with a fresh access token. For more information, see OAuth 2.0 client. **Note: if it's a rotatable token, add --rotatable to your script.**

Syntax	Abbreviated Syntax	Followed by a variable, if applicable	What it does
-retrytimeout	-rt	Timeout in seconds	The number of times, in seconds, to attempt to reconnect HTTP client calls if disconnected. Anaplan Connect retry behavior works as follows: 1. If there is a socket timeout (defined in the -httptimeout parameter), Anaplan Connect follows the socket timeout duration per retry attempt. 2. After the socket timeout interval, Anaplan Connect uses the value of the retrytimeout parameter. Each subsequent retry attempt multiplies the timeout value by 1.5. For example: if httptimeout is set to 60 and retrytimeout is set to 3, the first retry happens after 60+3 seconds. The second retry takes place 64.5 seconds later. The third retry takes place 66.75 seconds later. 3. The timeout has a maximum duration of 2 minutes (including the httptimeout and retrytimeout values combined). If not specified, the default is 3 seconds. The minimum value can be 3 and the maximum value 60. When using this parameter, specify it in your script before your *-execute* parameter. Also use this command directly after the –model command. Use directly after the -models or -model_id command. If you use this command in conjunction with -maxretrycount and -retrytimeout, use these commands in alphabetical order.
-quiet	-q		Show less detailed output

Appendix D: JDBC for Oracle, Access, and Excel

Provided you have the appropriate JDBC driver, you can link directly into Anaplan from databases such as Oracle, Access, MySQL, or from Excel. Both lists and data can be imported into Anaplan in this way, and when combined with a scheduler, can be updated on a regular basis automatically.

The command contains a single argument `–jdbproperties`, which provides the path to a properties file.

```
set Operation=-file "Anaplan_Demo_Sql" -jdbcproperties

</path/to/jdbc.properties>
```

The jdbc.properties file contains the connection details including the path to the database, username, password, and the query string.

Note: A properties file can contain only one query. If you have multiple queries, create a properties file for each query.

```
# JDBC Connection string (Oracle, Mysql, H2, etc.)
jdbc.connect.url= "jdbc:mysql://localhost:3306/apcustomer”

# JDBC login username
jdbc.username=user1

#JDBC login password
jdbc.password=dbpasswordforuser1

# (Optional) JDBC Fetch size, might not be applicable for all JDBC flavors.
jdbc.fetch.size=10

# Boolean flag, needs to be true if provided {jdbc.query} is a
# stored-procedure/function call.
# Defaults to false if no value provided, or invalid. jdbc.isStoredProcedure=false

# SQL query could be Select or a Stored-Procedure/Function call. In case of latter,
# require {jdbc.isStoredProcedure} above set to "true".
# NOTE: SQL query needs to be parametrized with ? placeholders and values for
# placeholders needs to be provided below in {jdbc.params}.
jdbc.query=” SELECT * FROM Anaplan where col1 = ? and col2 = ?”

# JDBC parameters for parametrized SQL query in {jdbc.query} above.

# NOTE: Number of parameters must equal to the number of parameters provided in {jdbc.query} above.
jdbc.params= 1234,ABC
```

Where:

-jdbc.connect.url	-jdbcurl is the location of your database instance. This location does not have to be on the local host if the machine connecting to the database has access to the network location. To connect to an ODBC source (no longer supported in JRE 8+), you must configure your URL as follows: "jdbc:odbc:location_of_instance".
-jdbc.username	The database username. If no user name or password is required by the source, omit the -jdbcpassword parameter.
-jdbc.password	The database password. If no user name or password is required by the source, omit the -jdbcpassword parameter.
-jdbc.fetch.size	Limits the number of rows that will be returned by the query.
-jdbc.isStoredProcedure	Boolean value that indicates whether the following statement (jdbc.query) is a stored procedure or function call.
-jdbc.query	Specifies the query type. -jdbc.query = "SELECT * FROM Anaplan where col1 = ? and col2 = ?" is the query to run.
-jdbc.params	JDBC parameters for parametrized SQL query in {jdbc.query} above. -jdbc.params = 1234,ABC

Appendix E: Import a JDBC Connection for a Microsoft SQL Server Database

An alternative to importing data into Anaplan from a file on the local host is to import data from a relational database using a Java Database Connectivity (JDBC) connection.

- The database can be any database that directly supports JDBC.

- Although you write Windows batch files (or Linux/Mac OS shell script files) for Anaplan Connect to run, Anaplan Connect itself is written in Java, and thus is well-suited for JDBC.

Preparation

If you want to create a connection to a Microsoft SQL Server database:

Copy a .jar file from the Microsoft SQL Server database server or client tools directory. If using Microsoft SQL Server 2008, the file name is sqljdbc41.jar.

Paste the .jar file into the lib subfolder of the Anaplan Connect installation.

Make sure you have the following information:

l UNC path to the Sql Server instance l Valid user name and password for the database login l A valid query to select the data you want brought into Anaplan.

For example, SELECT * FROM MYTABLE

An example batch file for an Import action through JDBC:

```
@echo off rem This example loads a source text file and runs an Anaplan import into a module.
rem For details of how to configure this script see doc\Anaplan Connect User Guide.doc set
AnaplanUser="Anaplan.User@anaplan.com:Password" set
WorkspaceId="8a8194884b27c72b014bf06a2b227f90" set
ModelId="CD9662D60CA84E9A871C1C5D061C7426" set Operation=-file "Anaplan_Demo_Sql"
jdbcproperties "C:\My Source.txt" > rem *** End of settings - Do not edit below this line ***
setlocal enableextensions enabledelayedexpansion || exit /b 1 cd %~dp0 if not
%AnaplanUser% == "" set Credentials=-user %AnaplanUser% set Command=.\AnaplanClient.bat
%Credentials% -workspace %WorkspaceId% -model %ModelId% %Operation% @echo %Command% cmd /c
%Command% pause set Operation=-service "https://api.anaplan.com" -auth
"https://us1a.app.anaplan.com" file "Anaplan_Demo_Sql" -jdbcproperties

</path/to/jdbc.properties>
```

The jdbc.properties file contains the connection details including the path to the database, username, password, and the query string.

Note: A properties file can contain only one query. If you have multiple queries, create a properties file for each query.

```
# JDBC Connection string (Oracle, Mysql, H2, etc.)
jdbc.connect.url=jdbc:sqlserver://localhost:1433;DatabaseName=my_database

# JDBC login username
jdbc.username=user1
```

```
#JDBC login password
jdbc.password=dbpasswordforuser1

# (Optional) JDBC Fetch size, might not be applicable for all JDBC flavors.

jdbc.fetch.size=10

# Boolean flag, needs to be true if provided {jdbc.query}
is a # stored-procedure/function call.
# Defaults to false if no value provided, or invalid.

jdbc.isStoredProcedure=false

# SQL query could be Select or a Stored-Procedure/Function call. In case
of latter, # require {jdbc.isStoredProcedure} above set to "true". # NOTE:
SQL query needs to be parametrized with ? placeholders and values for #
placeholders needs to be provided below in {jdbc.params}.

jdbc.query=SELECT * FROM Anaplan where col1 = ? and col2 = ?

# JDBC parameters for parametrized SQL query in {jdbc.query} above.
# NOTE: Number of parameters must equal to the number of parameters provided in {jdbc.query}
above. jdbc.params= 1234,ABC
```

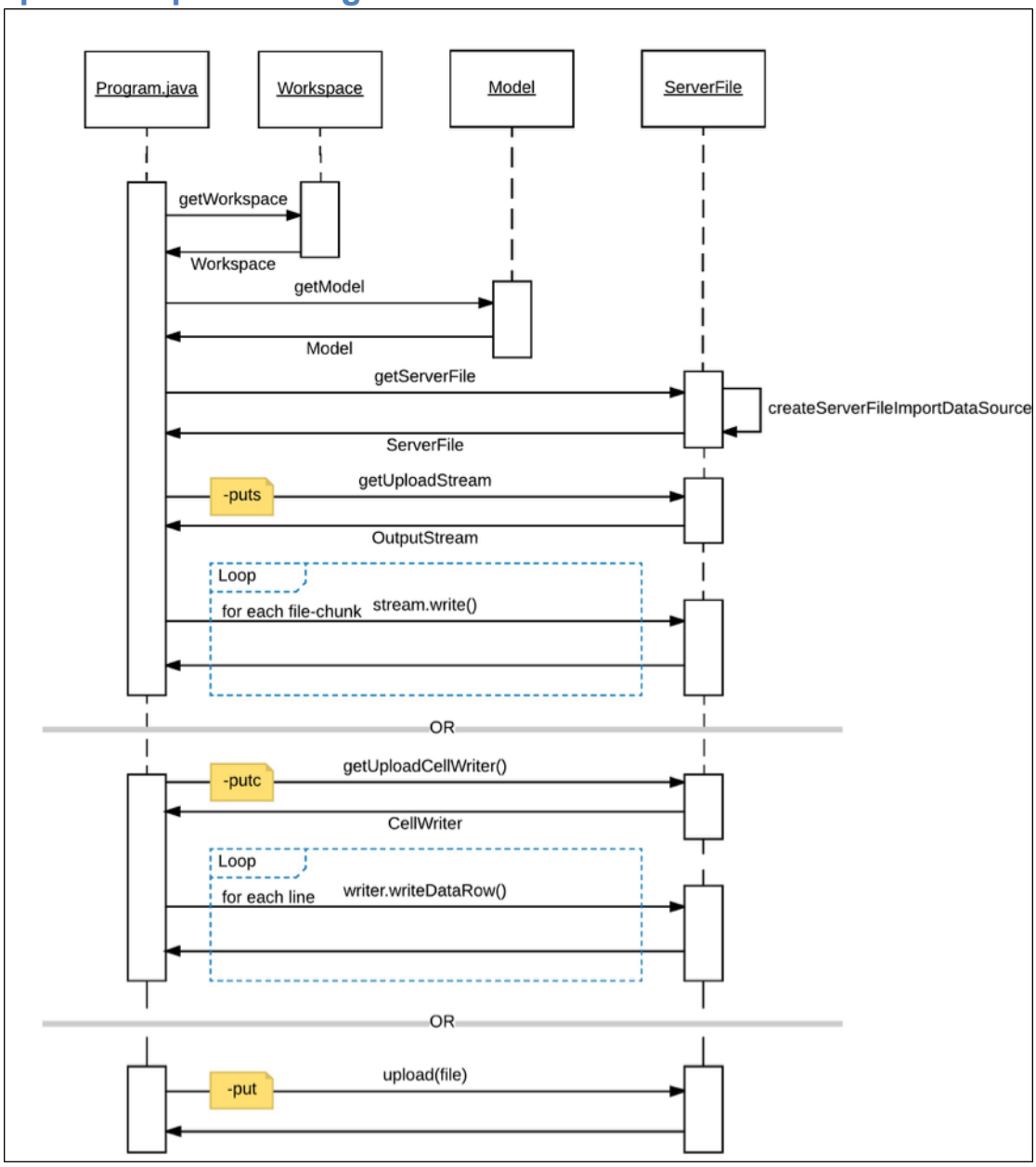
Query	Description
-jdbc.connect.url	The location of your database instance. If the connecting machine has network access, this location can be remote. To connect to an ODBC source, configure the URL as follows: jdbc:odbc:location_of_instance **Note**: If you want to use Anaplan Connect to import from an ODBC data source, Java 8 does not support the JDBC-ODBC Bridge.
-jdbc.username	The username for the JDBC database. If the source does not require a user name and password, do not include this parameter.

-jdbc.password	The password for the JDBC database. If the source does not require a user name and password, do not include this parameter.
-jdbc.fetch.size	Specifies the number of rows returned by the query.
-jdbc.isStoredProcedure	If this boolean value is true, the jdbc.query is defined as a stored procedure. If this value is false, the jdbc.query is defined as a function call.
-jdbc.query	Specifies the query type. For example: SELECT * FROM Anaplan where col1 = ? and col2 = ?
-jdbc.params	The JDBC parameters for the parametrized SQL query in the above -jdbc.query. For example: -jdbc.params=1234,ABC

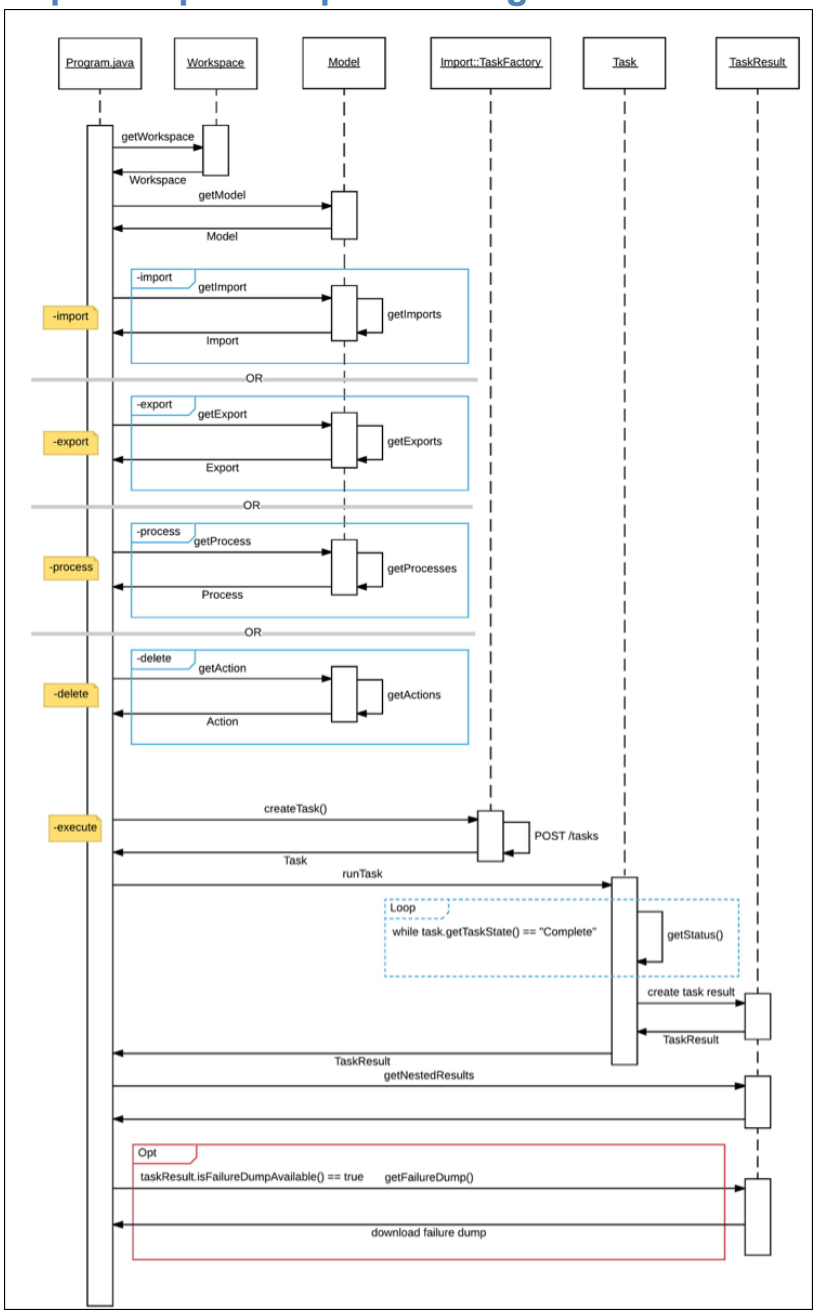
Appendix F: Sequence Diagrams

These diagrams illustrate the upload and import/export sequences used in Anaplan Connect.

Upload sequence diagram



Import/Export Sequence Diagram



Appendix G: Linux and MacOS Scripts

Linux and Mac OS example scripts are listed below.

Upload and Import Using a Proxy (Authenticated Proxy)

```
#!/bin/sh

#This example uploads a file and runs an import

AnaplanUser=username:password

WorkspaceId="workspaceid"

ModelId="modelid"

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

ActionName=""

ImportName="import action name"

ExportName=""

FileName="import data source name"

FilePath="import file location"

ChunkSize=1

OutputName=""

ErrorDump="error dump location"

ProxyUrl="proxyhost:proxyport"

ProxyUser=username:password


Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -via '${ProxyUrl}' -viauser ${ProxyUser}
workspace

${WorkspaceId} -model ${ModelId} -chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import

'${ImportName}' -execute -output '${ErrorDump}' "


#_____________________________ Do not edit below this line _____________________________


 if [ "${AnaplanUser}" ]; then

Credentials="-user

${AnaplanUser}" fi if [

"${CertPath}" ]; then

#Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias

${KeyStoreAlias}" # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA -certificate


#THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate

fi
```

```
echo cd "`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f AnaplanClient.sh ]; then
echo "Please ensure this script is in the same directory as AnaplanClient.sh."
>&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you have
executable permissions on AnaplanClient.sh." >&2 exit 1 fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec     /bin/sh     -c
"${Command}"
```

Upload and Import Using a Proxy (Unauthenticated Proxy)

```
#!/bin/sh
#This example uploads a file and runs an import

AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName="import action name"
ExportName=""
FileName="import data source name"
FilePath="import file location"
ChunkSize=1
OutputName=""
ErrorDump="error dump location"
ProxyUrl="proxyhost:proxyport"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -via '${ProxyUrl}' -workspace ${WorkspaceId}
model ${ModelId} -chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}'
execute output '${ErrorDump}' "

#_____________________________ Do not edit below this line _____________________________

if [ "${AnaplanUser}" ]; then
  Credentials="-user ${AnaplanUser}"
fi

if [ "${CertPath}" ]; then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
```

```
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c
"${Command}"
```

Delete Content from a Module (Basic Authentication)

```
#!/bin/sh
#This example deletes selected content from the module

AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName="delete action name"
ImportName=""
ExportName=""
FileName=""
FilePath=""
ChunkSize=""
OutputName=""
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
action '${ActionName}' -execute"
```

```
#_____________________________ Do not edit below this line _____________________________


if [ "${AnaplanUser}" ]; then
  Credentials="-user ${AnaplanUser}"
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same

  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fi


#Command="./AnaplanClient.sh ${Credentials} -workspace ${WorkspaceId} -model ${ModelId} ${Operation}"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec     /bin/sh     -c
"${Command}"
```

Delete Content from a Module (Certificate Authentication)

```
#!/bin/sh
#This example deletes selected content from the module
#AnaplanUser=integraqa@anaplan.com:Welcome1
CertPath="certificate location"
KeyStorePath="keystore location"
KeyStorePass="keystore password"
KeyStoreAlias="keystore alias"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName="delete action name"
ImportName=""
ExportName=""
FileName=""
FilePath=""
ChunkSize=""
OutputName=""
```

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
action '${ActionName}' -execute"

#_____________________________ Do not edit below this line _____________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user ${AnaplanUser}"
fi
if [ "${KeyStorePath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi

#Command="./AnaplanClient.sh ${Credentials} -workspace ${WorkspaceId} -model ${ModelId} ${Operation}"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec     /bin/sh     -c
"${Command}"
```

Delete Content from a Module (OAuth method)

```
#!/bin/sh
#This example exports a file
ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
```

```
ImportName=""

ExportName="exportactionname"

FileName="location for exported data"

FilePath=""

ChunkSize=""

OutputName=""

ErrorDump=""

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}

action '${ActionName}' -execute -get '${FileName}'"


#_____________________________ Do not edit below this line _____________________________

if [ "${ClientId}" ]; then
  Credentials="-oauth-client-id ${ClientId}"
fi echo cd "$(dirname "$0")" cd "$(dirname
"$0")"
if [ ! -f AnaplanClient.sh ]; then
  echo "Please ensure this script is in the same directory as AnaplanClient.sh." >&2
exit 1
elif [ ! -x AnaplanClient.sh ]; then
  echo "Please ensure you have executable permissions on AnaplanClient.sh." >&2
exit 1 fi
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec
/bin/sh -c "${Command}"
```

Delete Content from a Module (OAuth force register method)

```
#!/bin/sh

#This example exports a file

ClientId="clientid"

WorkspaceId="workspaceid"

ModelId="modelid"

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

ActionName=""

ImportName=""

ExportName="exportactionname"

FileName="location for exported data"

FilePath=""
```

```
ChunkSize=""
OutputName=""
ErrorDump=""

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} --forceRegister -workspace ${WorkspaceId}
model ${ModelId} -export '${ExportName}' -execute -get '${FileName}'"


#_______________________________ Do not edit below this line _____________________________


if [ "${ClientId}" ]; then
  Credentials="-oauth-client-id ${ClientId}"
fi
echo cd "$(dirname "$0")" cd
"$(dirname "$0")"
if [ ! -f AnaplanClient.sh ]; then
  echo "Please ensure this script is in the same directory as AnaplanClient.sh." >&2
exit 1
elif [ ! -x AnaplanClient.sh ]; then
  echo "Please ensure you have executable permissions on AnaplanClient.sh." >&2
exit 1 fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec
/bin/sh -c "${Command}"
```

Export (Basic Authentication)

```
#!/bin/sh
#This example exports a file
AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName=""
ExportName="exportactionname"
FileName="location for exported data"
FilePath=""
ChunkSize=""
OutputName=""
ErrorDump=""
```

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
export '${ExportName}' -execute -get '${FileName}'"

#_____________________________ Do not edit below this line _____________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user

${AnaplanUser}" fi if [

"${CertPath}" ]; then

 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}

keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA

certificate

      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate

fi

echo cd "`dirname "$0"`" cd

"`dirname "$0"`" if [ ! -f AnaplanClient.sh

]; then echo "Please ensure this script is

in the same directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo

"Please ensure you have executable permissions on

AnaplanClient.sh." >&2 exit 1 fic

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"

exec /bin/sh -c

"${Command}"
```

Export (Certificate Authentication)

```
#!/bin/sh

#This example uploads a file and runs an export


#AnaplanUser=integraqa@anaplan.com:Welcome1

CertPath="certificate location"

KeyStorePath="keystore location"

KeyStorePass="keystore password"

KeyStoreAlias="keystore alias"

WorkspaceId="workspaceid"

ModelId="modelid"
```

```
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName=""
ExportName="exportactionname"
FileName="location for exported data"
FilePath=""
ChunkSize=""
OutputName=""
ErrorDump=""

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
export '${ExportName}' -execute -get '${FileName}'"

#_____________________________ Do not edit below this line _____________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user
${AnaplanUser}" fi if [
"${KeyStorePath}" ]; then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA -certificate fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c
"${Command}"
```

Export (OAuth method)

```
#!/bin/sh
#This example exports a file
ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName=""
ExportName="exportactionname"
FileName="location for exported data"
FilePath=""
ChunkSize=""
OutputName=""
ErrorDump=""
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
export '${ExportName}' -execute -get '${FileName}'"

#_____________________________ Do not edit below this line _____________________________

if [ "${ClientId}" ]; then
   Credentials="-oauth-client-id ${ClientId}"
fi  if [ "${CertPath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA -certificate fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2   exit   1   elif   [   !   -x
AnaplanClient.sh ]; then echo "Please
ensure    you    have    executable
permissions on

AnaplanClient.sh." >&2 exit 1 fic
```

```
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Export (OAuth force register method)

```
#!/bin/sh
#This example exports a file
ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName=""
ExportName="exportactionname"
FileName="location for exported data"
FilePath=""
ChunkSize=""
OutputName=""
ErrorDump=""
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} --forceRegister -workspace ${WorkspaceId}
model ${ModelId} -export '${ExportName}' -execute -get '${FileName}'"

#_____________________________ Do not edit below this line _____________________________

if [ "${ClientId}" ]; then
   Credentials="-oauth-client-id ${ClientId}"
fi  if [ "${CertPath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA -certificate fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2   exit   1   elif   [   !   -x
AnaplanClient.sh ]; then echo "Please
ensure    you    have    executable
permissions on

AnaplanClient.sh." >&2 exit 1 fic
```

```
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Import (Basic Authentication)

```
#!/bin/sh
#This example uploads a file and runs an import

AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName="import action name"
ExportName=""
FileName="import data soruce name"
FilePath="import file location"
ChunkSize=1
OutputName=""
ErrorDump="error dump location"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}' -execute -output
'${ErrorDump}' "

#_____________________________ Do not edit below this line _____________________________ if
[ "${AnaplanUser}" ]; then
  Credentials="-user
${AnaplanUser}" fi if [
"${CertPath}" ]; then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f AnaplanClient.sh
]; then echo "Please ensure this script is
in the same directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
```

```
  >&2 exit 1
fi


Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c

"${Command}"
```

Import (Certificate Authentication)

```
#!/bin/sh
#This example uploads a file and runs an import

#AnaplanUser=integraqa@anaplan.com:Welcome1
CertPath="certificate location"
KeyStorePath="keystore location"
KeyStorePass="keystore password"
KeyStoreAlias="keystore alias"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName="import action name"
ExportName=""
FileName="import data soruce name"
FilePath="import file location"
ChunkSize=1
OutputName=""
ErrorDump="error dump location"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}' -execute -output
'${ErrorDump}' "


#_____________________________ Do not edit below this line _____________________________ if
[ "${AnaplanUser}" ]; then
  Credentials="-user
${AnaplanUser}" fi if [
"${CertPath}" ]; then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
 keystorealias
${KeyStoreAlias}"  # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate fi
echo cd "`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f AnaplanClient.sh ]; then  echo
"Please ensure this script is in the same directory as AnaplanClient.sh." >&2 exit 1
```

```
elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you have
  executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Import (OAuth method)

```
#!/bin/sh
#This example uploads a file and runs an import

ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName="import action name"
ExportName=""
FileName="import data soruce name"
FilePath="import file location"
ChunkSize=1
OutputName=""
ErrorDump="error dump location"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}' -execute -output
'${ErrorDump}' "

#_____________________________ Do not edit below this line _____________________________
  if [ "${ClientId}" ];
then
 Credentials="-oauth-client-id ${ClientId}"
 fi
  if [ "${CertPath}" ];
 then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate fi  echo cd "`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
directory as AnaplanClient.sh."
>&2   exit   1   elif   [   !   -x
AnaplanClient.sh   ];   then   echo
"Please ensure you have executable
```

```
permissions   on   AnaplanClient.sh."
>&2 exit 1 fi


Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}" (}"
```

Import (OAuth force register method)

```
#!/bin/sh
#This example uploads a file and runs an import

ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName="import action name"
ExportName=""
FileName="import data soruce name"
FilePath="import file location"
ChunkSize=1
OutputName=""
ErrorDump="error dump location"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} --forceRegister -workspace ${WorkspaceId}
model ${ModelId} chunksize ${ChunkSize} -file '${FileName}' -put ${FilePath} -import '${ImportName}'
execute -output '${ErrorDump}' "

#_____________________________ Do not edit below this line _____________________________
  if [ "${ClientId}" ];
then
 Credentials="-oauth-client-id ${ClientId}"
 fi
  if [ "${CertPath}" ];
 then
 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate fi  echo cd "`dirname "$0"`" cd "`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
directory as AnaplanClient.sh."
>&2   exit   1   elif   [   !   -x
AnaplanClient.sh   ];   then   echo
"Please ensure you have executable
```

```
permissions   on   AnaplanClient.sh."
>&2 exit 1 fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}" (}"
```

Run a Process (Basic Authentication)

```
#!/bin/sh
#This example runs a process action

AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName=""
ExportName=""
ProcessName="process action name"
FileName=""
FilePath=""
ChunkSize=""
OutputName=""
ImportDataSource1="import data source 1"
ImportFileName1="import file location 1"
ImportDataSource2="import data source 2"
ImportFileName2="import file location 2"
ErrorDump="error dump location"
ExportName1="export action name 1"
ExportName2="export action name 2"
FileName1="exported data location 1"
FileName2="exported data location 2"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
file '${ImportDataSource1}' -put '${ImportFileName1}' -file '${ImportDataSource2}' -put
'${ImportFileName2}' process '${ProcessName}' -execute -file '${ExportName1}' -get '${FileName1}' -file

'${ExportName2}' -get '${FileName2}' -output '${ErrorDump}'"

#_____________________________ Do not edit below this line _____________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user ${AnaplanUser}"
fi

if [ "${CertPath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
```

```
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fi


Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c

"${Command}"
```

Run a Process (Certificate Authentication)

```
#!/bin/sh
#This example uploads a file and runs an import
#AnaplanUser=integraqa@anaplan.com:Welcome1
CertPath="certificate location"
KeyStorePath="keystore location"
KeyStorePass="keystore password"
KeyStoreAlias="keystore alias"


WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ActionName=""
ImportName=""
ExportName=""
ProcessName="import export delete"
FileName=""
FilePath=""
ChunkSize=""
```

```
OutputName=""
ImportDataSource1="import data source 1"
ImportFileName1="import file location 1"
ImportDataSource2="import data source 2"
ImportFileName2="import file location 2"
ErrorDump="error dump location"
ExportName1="export action name 1"
ExportName2="export action name 2"
FileName1="exported data location 1"
FileName2="exported data location 2"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
file '${ImportDataSource1}' -put '${ImportFileName1}' -file '${ImportDataSource2}' -put
'${ImportFileName2}' process '${ProcessName}' -execute -file '${ExportName1}' -get '${FileName1}' -file

'${ExportName2}' -get '${FileName2}' -output '${ErrorDump}'"

#_________________________________ Do not edit below this line ___________________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user ${AnaplanUser}"
fi

if [ "${CertPath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fi

Command="./AnaplanClient.sh ${Credentials} ${Operation} "
```

```
/bin/echo "${Command}"
exec /bin/sh -c
"${Command}"
```

Run a process (OAuth method)

```
#!/bin/sh
#This example runs a process action


ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName=""
ExportName=""
ProcessName="process action name"
FileName=""
FilePath=""
ChunkSize=""
OutputName=""
ImportDataSource1="import data source 1"
ImportFileName1="import file location 1"
ImportDataSource2="import data source 2"
ImportFileName2="import file location 2"
ErrorDump="error dump location"
ExportName1="export action name 1"
ExportName2="export action name 2"
FileName1="exported data location 1"
FileName2="exported data location 2"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
file '${ImportDataSource1}' -put '${ImportFileName1}' -file '${ImportDataSource2}' -put
'${ImportFileName2}' process '${ProcessName}' -execute -file '${ExportName1}' -get '${FileName1}' -file
'${ExportName2}' -get '${FileName2}' -output '${ErrorDump}'"

#_____________________________ Do not edit below this line _____________________________
```

```
if [ "${ClientId}" ]; then
   Credentials="-oauth-client-id ${ClientId}" fi

if [ "${CertPath}" ]; then
 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}" }"
```

Run a process (OAuth force register method)

```
#!/bin/sh
#This example runs a process action

ClientId="clientid"
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
ProcessName="process action name"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName=""
ExportName=""
FilePath=""
FileName=""
ChunkSize=""
```

```
OutputName=""
ImportDataSource1="import data source 1"
ImportFileName1="import file location 1"
ImportDataSource2="import data source 2"
ImportFileName2="import file location 2"
ErrorDump="error dump location"
ExportName1="export action name 1"
ExportName2="export action name 2"
FileName1="exported data location 1"
FileName2="exported data location 2"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} --forceRegister -workspace ${WorkspaceId}
model ${ModelId} -file '${ImportDataSource1}' -put '${ImportFileName1}' -file '${ImportDataSource2}' put
'${ImportFileName2}' process '${ProcessName}' -execute -file '${ExportName1}' -get '${FileName1}' file
'${ExportName2}' -get '${FileName2}' -output '${ErrorDump}'"

#_________________________________ Do not edit below this line ___________________________________

if [ "${ClientId}" ]; then
   Credentials="-oauth-client-id ${ClientId}"
fi  if [ "${CertPath}" ]; then Credentials="-
keystore ${KeyStorePath} -keystorepass
${KeyStorePass} -keystorealias
${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}
keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA
certificate
      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate
fi

echo cd "`dirname "$0"`" cd
"`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."

  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fi
Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}" }"
```

Import with timeout defined

```
#!/bin/sh

#This example uploads a file and runs an import


AnaplanUser=username:password

WorkspaceId="workspaceid"

ModelId="modelid"

ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"

ActionName=""

ImportName="import action name"

ExportName=""

FileName="import data soruce name"

FilePath="import file location"

ChunkSize=1

RetryTimeout=3

MaxRetryCount=5

OutputName=""

ErrorDump="error dump location"


Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId} -

retrytimeout

'${RetryTimeout}' -maxretrycount '${MaxRetryCount}' chunksize ${ChunkSize} -file '${FileName}' -put

${FilePath} -import '${ImportName}'

-execute -output '${ErrorDump}' " #_____________________________ Do not edit below this

line _____________________________ if [ "${AnaplanUser}" ]; then

  Credentials="-user

${AnaplanUser}" fi if [

"${CertPath}" ]; then

 #Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}

keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA

certificate

      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate

fi

echo cd "`dirname "$0"`" cd

"`dirname "$0"`" if [ ! -f
```

```
AnaplanClient.sh ]; then echo "Please ensure this script is in the same
  directory as AnaplanClient.sh."
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."
  >&2 exit 1
fi


Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c
"${Command}"
```

Export with timeout defined

```
#!/bin/sh
#This example exports a file


AnaplanUser=username:password
WorkspaceId="workspaceid"
ModelId="modelid"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
ActionName=""
ImportName=""
ExportName="exportactionname"
FileName="location for exported data"
FilePath=""
ChunkSize=""
RetryTimeout=3
MaxRetryCount=5
OutputName=""
ErrorDump=""

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId} -
retrytimeout '${RetryTimeout}' -maxretrycount '${MaxRetryCount}' export
'${ExportName}' -execute -get '${FileName}'"


#_____________________________ Do not edit below this line _____________________________ if

[ "${AnaplanUser}" ]; then

  Credentials="-user

${AnaplanUser}" fi if [

"${CertPath}" ]; then

 Credentials="-keystore ${KeyStorePath} -keystorepass ${KeyStorePass} -keystorealias ${KeyStoreAlias}"
 #Credentials="-certificate ${CertPath} -keystore ${KeyStorePath} -keystorepass ${KeyStorePass}

keystorealias ${KeyStoreAlias}"   # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA

certificate

      # THIS IS ANOTHER APPROACH OF PROVIDING THE RAW CERTIFICATE VIA certificate

fi

echo cd "`dirname "$0"`" cd

"`dirname "$0"`" if [ ! -f

AnaplanClient.sh ]; then echo "Please ensure this script is in the same

  directory as AnaplanClient.sh."
```

```
  >&2 exit 1 elif [ ! -x AnaplanClient.sh ]; then echo "Please ensure you
have executable permissions on AnaplanClient.sh."

  >&2 exit 1
fic


Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c

"${Command}"
```

Display lists of workspaces, space allocated, and space used

```
#!/bin/sh

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -W" Credentials="-user ${AnaplanUser}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Display lists of models and space used.

```
#!/bin/sh

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -M" Credentials="-user ${AnaplanUser}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Get a complete list of modules and saved views within your model.

```
#!/bin/sh

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -views" Credentials="-user ${AnaplanUser}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Display data from saved views without using export actions (up to one million cells)

Single column format

```
#!/bin/sh

# This example provides view data in csv format

AnaplanUser=username:password
```

```
ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ModuleId="modulenameorid"

ViewId="viewnameorid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model ${ModelId}
module ${ModuleId} -vi ${ViewId} execute -get:csv_sc ${FilePath}" Credentials="-user ${AnaplanUser}"
echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Multi column format

```
#!/bin/sh

# This example provides view data in csv format

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ModuleId="modulenameorid"

ViewId="viewnameorid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId} -model '${ModelId}'
-module ${ModuleId} -execute -get:csv_ mc ${FilePath}"

Credentials="-user ${AnaplanUser}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Display data from saved views with parameterization (json format)

```
#!/bin/sh

# This example provides view data in json format

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ModuleName="modulenameorid"
```

```
ViewName="default"

Pages="pageDimensionName:dimensionMemberName"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl}
-workspace ${WorkspaceId}
-model ${ModelId} -mo ${ModuleName}
-view ${ViewName} pages \"${Pages}\"
-execute -get:json ${FilePath}

"credentials="-user ${AnaplanUser}" echo cd

"$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Display data from saved views with parameterization (csv format)

```
#!/bin/sh

# This example provides view data in csv format

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ModuleName="modulenameorid"

ViewName="default"

Pages="pageDimensionName:dimensionMemberName"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl}
-workspace ${WorkspaceId}
-model ${ModelId} -mo ${ModuleName}
-view ${ViewName} pages \"${Pages}\"
-execute -get:{csv |csv_sc|csv_mc} <local file path>
"credentials="-user ${AnaplanUser}"
echo cd "$(dirname "$0")" cd
"$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} " /bin/echo "${Command}" exec
/bin/sh -c "${Command}"
```

Note: same as previous command, but .csv format.

Display properties' details and show metrics for your model lists

```
#!/bin/sh

# This example list module list

AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"
```

```
FilePath="localfilesystempath"
ListId="listid"
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace ${WorkspaceId}
-model ${ModelId} -l ${ListId}
-get:csv ${FilePath}" Credentials="-user ${AnaplanUser}"  echo
cd "$(dirname "$0")" cd "$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c
"${Command}"
```

Display items from your model lists without using export actions (up to one million items)

With include all:

```
#!/bin/sh
# This example deletes selected content from the module AnaplanUser=username:password
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
WorkspaceId="workspaceid"
ModelId="modelid"
FilePath="localfilesystempath"
ListId="listid"
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace
${WorkspaceId} -model ${ModelId} -l ${ListId} -execute:all -get:csv
'${FilePath}'"
Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Without include all:

```
#!/bin/sh
# This example deletes selected content from the module
AnaplanUser=username:password
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://us1a.app.anaplan.com"
WorkspaceId="workspaceid"
ModelId="modelid"
FilePath="localfilesystempath"
ListId="listid"
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace
${WorkspaceId} -model ${ModelId} -l ${ListId} -execute -get:csv
```

```
'${FilePath}' "
Credentials="-user ${UserCredentials}" echo
cd "$(dirname "$0")" cd "$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

```
-workspace ${WorkspaceId} -model ${ModelId}
```

Add new list items without import actions

Syntax

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} list <list_id_or_name> -itemmappingproperty
<path_to_mapping_file> -execute -putItems:(csv|json|jdcb)
<path_to_local_file> -output <path_to_error_file>
```

Example without -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -list <list_id_or_name> -execute -putItems:(csv|json|jdcb)
<path_to_local_file>
Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Example with -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"
```

ItemMappingPropertyPath=”localfilesystempath”

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -list <list_id_or_name> -itemmappingproperty
${ItemMappingPropertyPath} -execute -puttItems:(csv|json|jdcb) <path_to_local_file>

Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"
```

```
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

A sample mapping property file

```
<local_column_name_1>=<anaplan_list_column_name_1>
<Local_column_name_2>=<anaplan_list_column_name_2>
```

Note: You can also include mapping information directly in the JDBC property file. Refer to the mapping file with item mapping parameter (-im or -itemmappingproperty). Here is an example:

```
jdbc.connect.url=jdbc:mysql://localhost:3316/retail_dat
a jdbc.username=root  jdbc.password=root_password
jdbc.fetch.size=100
jdbc.query=SELECT * FROM
sales_actuals a=east_coast_stores
b=sales_per_store "col 1"="
store$_sales"
```

```
                                                                    -workspace ${WorkspaceId} -model ${ModelId}
```

Update list items without import actions

Syntax

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl}                                                                    -
(list|l) <list_id_or_name> -itemmappingproperty <path_to_mapping_file> -execute
updateItems:(csv|json|jdcb) <path_to_local_file> -output <path_to_error_file>
```

Example without -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} "-list <list_id_or_name> -execute -updateItems:(csv|json|jdcb)
<path_to_local_file>

Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

Example with -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"

ItemMappingPropertyPath=”localfilesystempath”

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} "-list <list_id_or_name> -itemmappingproperty
```

```
${ItemMappingPropertyPath} -execute -updateItems:(csv|json|jdcb) <path_to_local_file> Credentials="-
user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

A sample mapping property file

```
<local_column_name_1>=<anaplan_list_column_name_1>
<Local_column_name_2>=<anaplan_list_column_name_2>
```

```
-workspace ${WorkspaceId} -model ${ModelId}
```

Upsert list items without import actions

Syntax

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -
(list|l) <list_id_or_name> -itemmappingproperty <path_to_mapping_file> -execute
upsertItems:(csv|json|jdcb) <path_to_local_file> -output <path_to_error_file>
```

Example without -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} "-list <list_id_or_name> -execute -upsertItems:(csv|json|jdcb)
<path_to_local_file>

Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

Example with -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"
```

ItemMappingPropertyPath=”localfilesystempath”

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -list <list_id_or_name> -itemmappingproperty
${ItemMappingPropertyPath} -execute -upsertItems:(csv|json|jdcb) <path_to_local_file>
```

```
Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"
Command="./AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

A sample mapping property file

```
<local_column_name_1>=<anaplan_list_column_name_1>
<Local_column_name_2>=<anaplan_list_column_name_2>
```

Delete list items without delete actions

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"

Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -l ${ListId} -list <list_id_or_name> -execute
deleteItems:(csv|json|jdcb) <path_to_local_file>
Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}"
```

Example with -itemmappingproperty parameter

```
#!/bin/sh

# This example deletes selected content from the module AnaplanUser=username:password

ServiceUrl="https://api.anaplan.com"

AuthUrl="https://us1a.app.anaplan.com"

WorkspaceId="workspaceid"

ModelId="modelid"

FilePath="localfilesystempath"

ListId="listid"
```

ItemMappingPropertyPath=”localfilesystempath”

```
Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace

${WorkspaceId} -model ${ModelId} -list <list_id_or_name> -itemmappingproperty
${ItemMappingPropertyPath} -execute -deleteItems:(csv|json|jdcb) <path_to_local_file>

Credentials="-user ${UserCredentials}" echo cd "$(dirname "$0")" cd "$(dirname "$0")"

Command="./AnaplanClient.sh ${Credentials} ${Operation} "

/bin/echo "${Command}" exec /bin/sh -c "${Command}
```

A sample mapping property file

```
<local_column_name_1>=<anaplan_list_column_name_1>
<local_column_name_2>=<anaplan_list_column_name_2>
```

Linux and Mac OS

```
#!/bin/sh
# For MAC OS : This example explains the usage of basic Authentication with Anaplan Connect.
# For details of how to configure this script visit
https://help.anaplan.com/anapedia/Content/Downloads/DL_Downloads.html



AnaplanUser="Username:Password"
WorkspaceId="Workspace Id"
ModelId="Model Id"
ServiceUrl="https://api.anaplan.com"
AuthUrl="https://auth.anaplan.com"
ExportName="Export Name"
FilePath="File Path"


Operation="-debug -service ${ServiceUrl} -auth ${AuthUrl} -workspace_id ${WorkspaceId} -model_id
${ModelId} -export '${ExportName}' -execute -get '${FilePath}' "


#_______________________________ Do not edit below this line _________________________________


if [ "${AnaplanUser}" ]; then
Credentials="-user ${AnaplanUser}"
fi
echo cd "`dirname "$0"`" cd
"`dirname "$0"`"
if [ ! -f ../AnaplanClient.sh ]; then
  echo "Please ensure this script is in the same directory as AnaplanClient.sh." >&2
exit 1
elif [ ! -x ../AnaplanClient.sh ]; then
  echo "Please ensure you have executable permissions on AnaplanClient.sh."
>&2   exit 1 fi


Command="../AnaplanClient.sh ${Credentials} ${Operation} "
/bin/echo "${Command}"
exec /bin/sh -c "${Command}"
```

Run with Model ID and Workspace ID, basic Auth method

Windows

```
@echo off rem For Windows OS : This example explains the usage of basic Authentication with
Anaplan Connect.
rem For details of how to configure this script visit
https://help.anaplan.com/anapedia/Content/Downloads/DL_Downloads.html
```

```
set
AnaplanUser="Username:Password"
set WorkspaceId="My Workspace" set
ModelId="My Model"
set
ServiceUrl="https://api.anaplan.com"
set AuthUrl="https://auth.anaplan.com"
set FilePath="File Path" set
ExportName="Export Action"


set Operation=-debug -service %ServiceUrl% -auth %AuthUrl% -workspace_id %WorkspaceId% -model_id
%ModelId% -export %ExportName% -execute -get %FilePath%


rem *** End of settings - Do not edit below this line ***


setlocal enableextensions enabledelayedexpansion || exit /b 1
cd %~dp0 if not %AnaplanUser% == "" set Credentials=-user
%AnaplanUser% set Command=..\AnaplanClient.bat %Credentials%
%Operation%
@echo  %Command%
cmd /c %Command%
pause
```