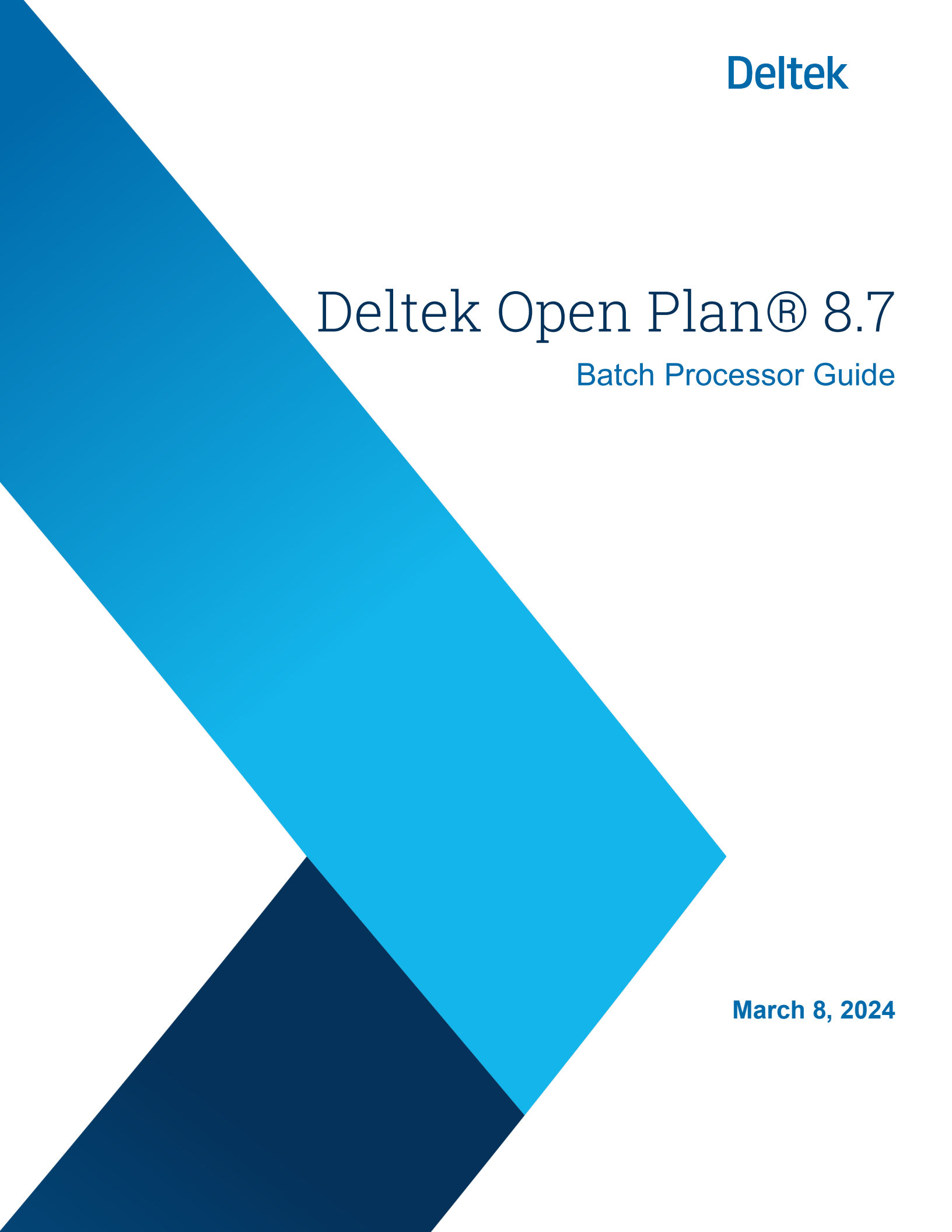




Deltek

Deltek Open Plan® 8.7

Batch Processor Guide

March 8, 2024



While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published March 2024.

© Deltek, Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties.

All trademarks are the property of their respective owners.

Contents

Overview	1
Run the Open Plan Multi-Instance Batch Processor	2
Set Up the Console Batch File	3
Set Up the Process Batch File	5
Log Files	6
Console Application Log File	6
Batch Log Files	6
Process Batch Supported Commands	8
Commands	8
APPLY	8
ATTACHVIEW	8
AUTOCLOSEANCILLARY	8
BACKUP	9
BACKUPDIR	9
BASELINEUPDATE	9
CC COSTCALCULATIONS COSTCALCS	10
CLOSE SHUT	10
DROPPROJECT	10
END QUIT EXIT	11
EXPORT	11
GLOBALBACKUP	11
IMPORT	12
MINIMIZE MINIMISE	12
ONERROR	12
OPEN OPENFILE	13
OPENEX OPENMP	13
PC PROGCALCS PROGRESSCALCULATIONS	13
REM REMARK //	14
RESOURCE REPLACE	14
RS RESOURCESCHEDULE SCHEDULE	15
SAVE	15
SAVEAS	16
SHOWLOG	16

TA TIMEANALYSE TIMEANALYZE	16
TIMENOW	16
Open Plan Internal Batch Processor	18
<i>Open Plan API</i>	20
OPProject Object	20
OPApplication Object	20
Methods	20
Properties	20

Overview

The Open Plan Batch (OPBatch) utility supports running multiple instances of Open Plan simultaneously and adds additional commands for increased batch processing control and error handling.

The Multi-Instance Batch processor consists of a console application (**OPPMultiInstanceBatch.exe**) that launches multiple instances of Open Plan using an internal implementation of OPBatch-type functionality within the Open Plan application (OPP) natively.

Two batch files are used in the process.

- The first batch file is referred to as the “Console Batch file”. Each line in this file supplies information for each instance of OPP that will be launched (user name, password, project, batch sequence number, and batch file to be executed for that instance). The console app (a Microsoft Windows Command Prompt window, which is created by the **OPPMultiInstanceBatch.exe**) formats this information using command-line switches and passes this command line as it creates a new Open Plan process.
- The OPP process then executes the batch file that has been assigned to it. This batch file contains OPBatch-type instructions and is referred to below as the “Process Batch file.” A single batch file may be used for all instances of Open Plan that are launched, or if desired, the batch file may be unique for each instance.

Run the Open Plan Multi-Instance Batch Processor

From a Command Prompt (or using **Windows » Start Menu » Run** command), you can invoke the Multi-Instance Batch Processor as follows:

Syntax: OPPMultiInstanceBatch.exe OpenPlanExePath ConsoleBatchFile StartOfRange
[DefaultProcessBatchFile] [ConsoleLogFilePath]

- **OpenPlanExePath** – This parameter is mandatory and points to a valid, fully-qualified Open Plan executable file name that will be used to launch all instances. If there are any spaces in the executable's path, the argument should be enclosed in double quotation marks.
- **ConsoleBatchFile** – This parameter is mandatory and points to a valid, fully-qualified text file name. If there are any spaces in the file name, the argument should be enclosed in double quotation marks.
- **StartOfRange** – This parameter is mandatory and must be a number between 1 and 100,000 (0 is an invalid number). The StartOfRange parameter uniquely identifies the instances of OPP launched from within a console application. The range numbers used by a given instance of the console application must be unique to that instance.
- **[DefaultProcessBatchFile]** – This parameter is optional, but if the last parameter, [ConsoleLogFilePath], is specified in the command line arguments then the [DefaultProcessBatchFile] parameter must be supplied as well.

The [DefaultProcessBatchFile] argument may be specified as "None," which means that the actual Process Batch File to be used for each batch will be specified within the Console Batch File.

- **[ConsoleLogFilePath]** – This parameter is optional and points to a valid, fully-qualified path where the batch log file will be written. If there are any spaces in the file name, the argument should be enclosed in double quotation marks.

Set Up the Console Batch File

The Console Batch File is mandatory. It contains one line of text for each instance of OPP to be launched. Values are comma-delimited and quotation marks are not used. The format for each line of text is as follows:

Syntax: UserName, Password, Project, LogFileSuffix, ProcessBatchFile

- **UserName** – The Console application uses the specified UserName to log in to Open Plan. The parameter can be set to a specific UserName or it may be set to “Authentication” (without the quotes) if you wish to log in via Windows Authentication. Using “Authentication,” assumes that a user login will be set up for the current user (using the EPM Security Administrator tool) with appropriate access rights for the project(s) and task(s) processed in the Process Batch File.
- **Password** – The Console application uses the specified Password to log in to Open Plan. The parameter can be set to a specific User’s Password or it may be set to “Authentication” (without the quotes) if you wish to log in via Windows Authentication. Using “Authentication,” assumes that a user login will be set up for the current user (using the EPM Security Administrator tool) with appropriate access rights for the project(s) and task(s) processed in the Process Batch File.
- **Project** – The Project argument must always be specified, but a value of NONE indicates that there is no default project for this instance. It means that the “OPEN @PROJ” syntax cannot be used within the Process Batch File (see comments about the OPEN command in the “Setting up The Process Batch File” section below.)
- **LogFileSuffix** – The LogFileSuffix argument specifies an identifier that will be appended to the root name of an OPP log file so that log files created during the batch process for different batches may be differentiated from one another.

For example, if “GMK” (without the quotes) is passed as a value, the opwin.log file would be named opwin_GMK.log. An underscore is added by Open Plan to separate the suffix from the root file name; however, including an underscore in the LogFileSuffix is permitted. The LogFileSuffix argument must be supplied, but if a value of “DEFAULT” (without the quotes) is passed, Open Plan will form its suffix based on the starting sequence number (with a starting sequence number of 1000, the suffix generated for the first batch processed will be 1000; the second, 1001, and so on and so forth).

- **ProcessBatchFile** – The ProcessBatchFile argument is optional only if a valid argument has been specified for the [DefaultProcessBatchFile] parameter when the console application was launched.

The [DefaultProcessBatchFile] parameter specified on the Console Application’s command line is an optional argument as well. If all Open Plan instances will be performing the same steps (that is, using the same batch file), setting this parameter on the command line avoids the inconvenience of repeating the same batch file name for each Open Plan instance within the Console Batch File.

Setting the command line parameter serves as a default, allowing you to mix and match Process Batch Files within the Console Batch File (that is, omitting the ProcessBatchFile argument from an Open Plan instance within the Console Batch File will revert to the [DefaultProcessBatchFile] specified on the command line).

Examples:

1. USER1, PASSWORD1, CLEAN , DEFAULT, C:\MyStuff\BatchSteps.txt
 - Log in as “USER1/PASSWORD1” with CLEAN as the default project.

- DEFAULT: The batch sequence number (relative position of the batch or line number) will be used to create the log file “suffix” in OPP.
- The supplied ProcessBatchFile overrides any default that was specified when the Console App was launched.

2. AUTHENTICATION, AUTHENTICATION, NONE, GMK

- Log in using Windows Authentication.
- NONE: There is no default project.
- “GMK” will be used as the suffix when log files are created in OPP.
- Since the optional fifth argument is not specified, the [DefaultProcessBatchFile] will be used (an error will occur if the [DefaultProcessBatchFile] has not been specified or is invalid).

Set Up the Process Batch File

The ProcessBatchFile contains the batch steps to be executed by Open Plan. The syntax for the command lines is as follows:

- The Open Plan Batch Processor accepts commands in a simple structured form. The Batch Processor is case insensitive (that is, it accepts both upper and lower case characters).
- Each command is on its own line.
- There is always a space between the command (even the “//” command) and the first argument (if applicable).
- Arguments (when required) are separated by spaces. If a space occurs within the argument, the space should be replaced with a “tilde” (~) character.

Example:

The following is invalid:

```
IMPORT Activity Data C:\ACTDATA.CSV
```

It should be entered as:

```
IMPORT Activity~Data C:\ACTDATA.CSV
```

OR

```
IMPORT "Activity Data" C:\ACTDATA.CSV
```

Log Files

Console Application Log File

The Console App creates a single log file (OPPMultiInstConsole.log) in the directory where the Console Application executable, OPPMultiInstanceBatch.exe, is located (default location) or where the file is specified via the [ConsoleLogFilePath] argument. This log file contains both informational and error messages (the log level is currently hard-coded to display all messages) related to the processing of the Console Batch file. Any errors that occur prior to the log file being created display in the Command Prompt window.

Batch Log Files

A batch log file is generated for each instance that is launched from the Console Batch File and is written to Open Plan's User Directory. Instead of creating a separate summary log file containing the errors and warnings from the individual Console Batch instances, the errors and warnings are summarized and merged up to the Console Application Log file when all Open Plan instances launched from that instance of the console app have completed.

The information content appearing in the Console Batch Log files will not be merged up, only exception information. Each error line will be contained within brackets, and each warning line within curly braces ({}), followed by an error explanation:

Examples:

[1,CLEAN,Time Analysis,6,Failure] A loop was found that prevented Time Analysis from completing.
{2,PHARMACY,SHOWLOG,5,Warning} The ShowLog command is ignored for multi-instance batch mode.

Note: This information displays below any output generated as part of the Console Application execution.

This is the current list of constants. They may be re-ordered as required.

Note: Error numbers, when represented as strings, are limited to two characters (including the negative sign), allowing values from -9 to 99.

```
const int MIB_SUCCESS_UNKNOWN = -1;
//      No commands were executed in console or process batch file.
const int MIB_BATCH_SUCCEEDED = 0;
const int MIB_WARNING_REGISTRY_DELETE = 1;
//      Could not delete registry keys (does not cause any process to
fail)
const int MIB_WARNING_IN_PROCESS_BATCH_SYNTAX = 2;
const int MIB_WARNING_IN_PROCESS_BATCH_EXECUTION = 3;
//      Allow some room between the last warning message and the first
error message in case you want to add additional warnings.
```

```
const int MIB_FAILED = 15;
const int MIB_FAILURE_IN_CONSOLE_BATCH_SYNTAX = 16;
const int MIB_FAILURE_IN_CONSOLE_BATCH_EXECUTION = 17;
const int MIB_FAILURE_IN_PROCESS_BATCH_SYNTAX = 18;
const int MIB_FAILURE_IN_PROCESS_BATCH_EXECUTION = 19;
const int MIB_FAILURE_IN_COMMAND_LINE_SYNTAX = 20;
//      either console or process
const int MIB_FAILURE_TO_OPEN_PROCESS_BATCH_FILE = 21;
const int MIB_FAILURE_TO_FIND_PROCESS_BATCH_FILE = 22;
const int MIB_START_OF_RANGE_INVALID = 23;
const int MIB_ABNORMAL_TERMINATION = 24;
const int MIB_FAILURE_TO_CREATE_MUTEX = 25;
const int MIB_FAILURE_TO_CREATE_CONSOLE_LOGFILE = 26;
const int MIB_OPP_PATH_INVALID = 27;
const int MIB_CONSOLE_BATCH_PATH_INVALID = 28;
const int MIB_DEFAULT_PROCESS_BATCH_PATH_INVALID = 29;
const int MIB_FAILURE_TO_OPEN_CONSOLE_BATCH_FILE = 30;
const int MIB_FAILURE_IN_REGISTRY_IO = 31;
const int MIB_FAILURE_IN_FILE_IO = 32;
const int MIB_FAILURE_TO_LOGIN = 33;
const int MIB_MUTEX_TIMEOUT = 34;
```

The normal Open Plan session log file names used during a single-instance of an OPP session (for example, opwin.log, and opp_ta.log) are not suitable for multi-instance batch processing because each OPP instance must have exclusive access to its log files.

Note: The default multi-instance nomenclature is to append an underscore to the root part of the standard log file name, followed by the word “User” + the batch sequence number. This is the log file’s “suffix”. If the batch sequence number = 1, the opwin.log file would be named “opwin_User1.log” (the suffix is “User1”).

In the event that OPP determines, for some reason, that it cannot use the standard suffix (which normally should not happen), it increments the user number until it is successful.

Note: A custom suffix may be specified for each instance of OPP launched by the console application via the “log file suffix” parameter.

All processes (for example, T/A and R/S) performed during the running of a batch instance (across projects and multiple incarnations of the same process run on the same project) append their log information to the appropriate Open Plan process log file.

In addition to the log files normally created by OPP during a session, a new log file called Batch_Suffix.log is created to display information related to the processing of the Process Batch file.

Process Batch Supported Commands

All major commands currently existing in OPBatch are supported except for “Generate” because generating a VB script from the Process Batch file is unnecessary.

Note: It is unnecessary to include the SILENTMODE command in a Process Batch File because this is done automatically when the batch is processed. If the command is encountered, however, it does no harm (it’s just ignored).

The following commands are not currently supported because they are unnecessary for the purposes of the batch process (and in some cases, they could override information that has been supplied for a particular OPP instance):

- LOGDIR
- LOGIN
- EXIT
- SHOWLOG
- SILENTMODE (automatically set)
- ACTIVATEVIEW

Commands

APPLY

Syntax: APPLY GLOBALEDIT | BATCHGLOBALEDIT Name

Description: This command can be used to apply previously defined Global Edits or Batch Global Edits on the current project.

ATTACHVIEW

Syntax: ATTACHVIEW ViewName

Description: Attaches a view from the Open Plan **Views** folder within the **Open Plan Library** to the open project.

AUTOCLOSEANCILLARY

Syntax: AUTOCLOSEANCILLARY [True/False]

Description: This command can be used to prevent ancillary files (calendar, resource, and codes) opened with a project from being closed when the project is closed. This is useful when customers are working on a group of projects that have common ancillary files. Calling AUTOCLOSEANCILLARY FALSE can keep the same ancillary files from being opened and closed repeatedly and doing so improves performance (the series of batch operations completes more quickly). When the customers are finished processing their projects, AUTOCLOSEANCILLARY TRUE should be called so that the ancillary files will not remain open.

Example:

```
AUTOCLOSEANCILLARY FALSE
```

(prevents ancillary files from being closed when the project is closed).

BACKUP

Syntax: BACKUP SaveBaseline [True/False] BackupFileName

Description: Backup the currently opened project. A directory must have been previously specified with the BACKUPDIR command. If none of the parameters are defined, then it is assumed that the baselines will be included and the backup name will be the project name.

Example:

```
BACKUP TRUE CLEAN09MAR04
```

Valid parameters are:

SaveBaseline = Include the current baselines in the backup. Valid parameters are TRUE | False.

BackupFileName = Name of the backup file. If no parameter exists, then the current project name is used.

Note: Baselines are always included with OPP 3.x, despite the True/False setting here.

BACKUPDIR

Syntax: BACKUPDIR <backup directory>

Description: Sets the directory for the location of any files with the BACKUP command. The command can be placed once at the beginning of the script file and it will apply to all backups. The command can also be placed at any other location in the script file and any BACKUP commands following it will point to that directory.

Example:

```
BACKUPDIR C:\OPBatch\Backups
```

BASELINEUPDATE

Syntax: BASELINEUPDATE [BaselineName] [Filter] [1 | 2 | 3 | 4 | 5 | 6 | 7]

Description: This command updates current baselines in the currently selected project. You need to define up to seven parameters need to be defined. If any of the parameters are missing, they are set to TRUE by default.

Valid Parameters are:

BaselineName - Name of baseline to update, for example, PMB

Filter - Filter name to use. If no filter is going to be used, use NONE.

1 = In-progress activities are included (TRUE/FALSE)

2 = Completed activities are included (TRUE/FALSE)

3 = Activities existing in baseline are updated (TRUE/FALSE)

4 = Roll-up baseline info to the parent level (TRUE/FALSE)

5 = Delete activities no longer in the schedule (TRUE/FALSE)

6 = Use external baseline with same name as master baseline (TRUE/FALSE)

7 = Use remaining quantity for planned activities (TRUE/FALSE)

Example:

```
BASELINEUPDATE PMB NONE TRUE TRUE FALSE TRUE FALSE FALSE FALSE
```

CC | COSTCALCULATIONS | COSTCALCS

Syntax: CC [1 | 2 | 3 | 4 | 5 | 6 | 7]

Description: This command performs Cost Calculations on the open project. You may define up to 7 parameters. In the absence of any argument, the existing value on the Project Directory for the setting in question is used. The value USE_CURRENT allows you to keep the existing value of a setting while changing values for the parameters that follow:

- 1 = Calculate Budget Cost (True/False)
- 2 = Calculate Remaining Cost (True/False)
- 3 = Calculate Actual Cost (True/False)
- 4 = Calculate Earned Values (True/False)
- 5 = Based on: (Early, Late, Scheduled, Baseline)
- 6 = Use Resource Cost Escalation (True/False)
- 7 = Include child activity values in subproject costs (True, False)

Example:

```
CC TRUE TRUE FALSE USE_CURRENT SCHEDULED FALSE TRUE
```

CLOSE | SHUT

Syntax: CLOSE | SHUT [NOSAVE]

Description: This command closes an Open Plan project without saving its data. If a user would like data to be saved, the SAVE command must be included prior to executing the CLOSE command. If the NOSAVE parameter is specified, internally Open Plan calls the automation method ShutWOsave instead.

Note: If a project's data is modified, you must explicitly call SAVE to save your changes or use the CLOSE NOSAVE command to allow him to close the project w/o saving changes. Otherwise, a Save dialog will display and get in the way of the process since it's a modal dialog.

DROPPROJECT

Syntax: DROPPROJECT [ProjectName] [ProjectName...]

Description: This command deletes one or more projects. Ancillary files are not affected. If multiple projects are referenced and an error occurs, the process is aborted and the remaining projects are not deleted.

Valid Parameters are:

ProjectName - Name of project to delete, for example, CLEAN.

Example:

```
DROPPROJECT CLEAN CONSTR MOVIE SHIP
```

END | QUIT | EXIT

Syntax: END | QUIT | EXIT

Description: This command stops the Batch Processor, closes all files, and disconnects from Open Plan.

EXPORT

Syntax: EXPORT ScriptName FileName

Description: This command executes a general export script against the open project. The data is saved to the specified file name. If a project is not open, no action is performed.

Note: If the Export ScriptName or FileName contains spaces, each space should be replaced with a tilde (~) in the batch script file.

GLOBALBACKUP

Syntax: GLOBALBACKUP [FILTER] [NOTIMESTAMP]

Description: Backs up all of the projects in the Open Plan database or only those that you want based on the filter expression that you specify. You must have already specified a backup directory using the BACKUPDIR command.

- If the optional NOTIMESTAMP parameter is specified, only the project name is used. The format is "PROJECT NAME_DYYYYMMDDTHHMMSS.bk3".

Example:

CLEAN.bk3 (NOTIMESTAMP option)

CLEAN_D20100902T154500.bk3 (default)

The default behavior is to include the system time at the time the project backup is executed in the backup file name.

- If the optional FILTER parameter is specified, use the format FILTER <space> <"EXPRESSION or Named Filter on the Project table">.

Example

FILTER "INLIST (DIR_ID, 'CLEAN', 'CONSTR')"

You may use a date to compare to the Last Update date. You may also use both optional parameters NOTIMESTAMP and FILTER. In the following example, the backup command will back up any project with a Last Update date equal to or greater than 01/01/2021.

Example

FILTER 'LASTUPDATE >= {01/01/2021}' NOTIMESTAMP

In the following example, the backup command will back up any project with a non-blank Last Update date:

Example

FILTER "LASTUPDATE <> {}"

Note: The Last Update date uses a four-digit year format and is consistent with the Windows regional format settings.

IMPORT

Syntax: IMPORT ScriptName FileName [Active]

Description: This command executes a general import script. Data is read from the file name specified. If the optional **Active** parameter is specified, the data is loaded into the open project. Otherwise, data is loaded into a new untitled project.

Note: If the Import ScriptName or FileName contains spaces, each space should be replaced with a tilde (~) in the batch script file.

The argument following IMPORT is the specification name. The specification name (not the file name) must be in the Transfer.dat file. Here is an example of a Transfer.dat file where **MYIMPORT** is the specification name:

```
IMPORT csv MYIMPORT
DATE_FORMAT %M/%D/%Y
TABLE ACT
UPDATE_ONLY
FIELD ACT_ID
FIELD PROGTYPE
FIELD ASDATE
FIELD AFDATE
FIELD XFDATE
FIELD PPC
FIELD USER_CHR01
```

The csv after IMPORT tells you that the input source is a .csv file.

MINIMIZE | MINIMISE

Syntax: MINIMIZE | MINIMISE

Description: If a project is open, the current project window is minimized within Open Plan. If the project is not open, this command minimizes the Open Plan application.

ONERROR

Syntax: ONERROR [LABEL]

Description: This command allows execution to continue at a certain point in the batch script if an error occurs. It takes an optional argument, LABEL, which should be jumped to if an error occurs. The label should be positioned on a line of its own, followed by a colon (the colon is not included in the label name following the ONERROR command). If ONERROR is specified without the LABEL argument, the previously referenced LABEL is cleared.

Example:

```
OPEN @PROJ
ONERROR FAILED
TA
RS
SAVE
FAILED:
CLOSE NOSAVE
```

OPEN | OPENFILE

Syntax: OPEN ProjectName [Exclusive | Shared | ReadOnly]

Description: This command opens the specified project. The default is set to open in Exclusive mode, but this can be changed with the optional parameter.

Note: For the “OPEN” command (and its variations, OPENEX), the project name argument may be replaced with @PROJ. This allows the project name specified in the Console Batch file to be used instead of a fixed project name.

In addition, unlike the classic OPBATCH, the default startup view is automatically suppressed when the project is opened.

OPENEX | OPENMP

Syntax: OPENEX ProjectName [Exclusive | Shared | ReadOnly] [Yes | True]

Description: This command opens the specified project. The default is set to open in Exclusive mode, but this can be changed with the optional parameter to either Shared mode or Read Only mode. The default is to not open any of the external subprojects, but this can be change with the optional parameter to open all of the external subprojects.

PC | PROGCALCS | PROGRESSCALCULATIONS

Syntax: PC [1 2 3 4 5 6 7 8 9 10 11 12]

Description: This command performs Progress Calculations on the open project. You may define up to 13 parameters. In the absence of any argument, the existing value on the Project Directory for the setting in question is used. The USE_CURRENT value allows you to keep the existing value of a setting while changing values for the parameters that follow.

```
//      1)      Filter Name (or "NONE")
//      2)      Auto Progress Activity (True/False)
//      3)      Based on: (Early,Late,Scheduled,Baseline)
//      4)      Complete if finished before T/N: (True/False)
//      5)      In Progress if started before T/N: (True,False)
//      6)      Progress Type: (Remaining,Elapsed,Percent,Expected)
//      7)      Update resource remaining quantity (True/False)
```

Process Batch Supported Commands

```
//      8)          Update resource actual quantity (True/False)
//      9)          Update Physical Percent Complete (True/False)
//      10)         Base PPC on EVT (True/False)
//      11)         Progress Start
//      12)         Progress Finish
```

Example:

```
PC NONE TRUE EARLY USE_CURRENT USE_CURRENT TRUE TRUE FALSE TRUE FALSE 2/13/03
5/30/03
```

REM | REMARK | //

Syntax: REM | REMARK | //

Description: The text following the command is ignored.

Note: Like other commands, there should be at least one space between the command and any text that follows.

Example:

```
// These are my comments (NOTE: '//'These are my comments' is invalid because
there is no space between the '/' and 'These'.)

REM These are my comments
```

RESCERCEREPLACE

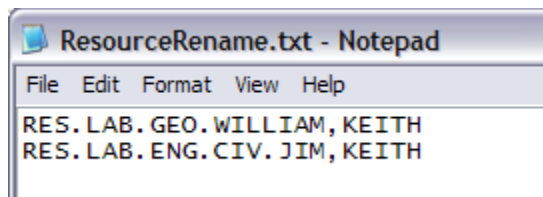
Syntax: RESCERCEREPLACE <path to replacement CSV or text file>

Description: Replace resource in the active project. The resource replace depends on a text file that defines the old and new resource name.

Example:

```
RESCERCEREPLACE C:\OPBatch\ResourceRename.txt
```

The comma-delimited file is in the following format:



The first column is the old resource and the second column is the new resource.

Note: If the path to the CSV or text file contains spaces, each space should be replaced with a tilde (~) in the batch script file.

RS | RESOURCESCHEDULE | SCHEDULE

Syntax: RS [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13]

Description: This command performs a Resource Schedule on the open project. All of the Parameters are optional and, if they are not specified, any information stored currently in the Project Directory for a particular value is used (except if any of the PRIORITY fields are specified, none of the values for any of the three priorities is pulled from the Project Directory).

- 1 = Scheduling Method (Time/Resource)
- 2 = Consider Usage on Higher Priority Projects (if applicable) (True/False)
- 3 = Ignore Actual Dates (True/False)
- 4 = In-progress Priority (True/False)
- 5 = Hard Zeroes (True/False)
- 6 = Smoothing (True/False)
- 7 = Scheduling Interval (Days/Hours/2H/4H/6H/8H/12H/Days/Weeks/Months)

Note: Resource Selection for scheduling is not workable as an option.

- 8 = Force Overloaded Activities to Late Dates (True/False)
- 9 = Try Alternate Resource to Avoid Delay (True/False)
- 10 = Limit Reprofileing to Original Level (True/False)

Note: In addition to allowing field names from the activity table, the priority fields below also allow the use of three Symbolic identifiers. Because these identifiers are not compatible with batch processing, Deltek has defined three constants:

- **\$\$None** — <None> in the database
- **\$\$Remaining_Float** — <Remaining Float> in the database
- **\$\$Hierarchical_Priority_Field** — <Hierarchical Priority Field> in the database

- 11 = First Priority
- 12 = Second Priority
- 13 = Third Priority

The value "USE_CURRENT" allows you to have a "placeholder" for a certain parameter without changing the Project Directory value associated with that parameter.

SAVE

Syntax: SAVE

Description: If the open project is open in Exclusive mode, the data is saved. If the project is open in any other mode, this command is ignored.

Note: Calling Save on a project that is not open in Exclusive mode currently causes a FATAL error to occur.

SAVEAS

Syntax: SAVEAS ProjectName [NODATA]

Description: The current open project is saved to a new project name. Any existing project of the same name is overwritten. The optional NODATA parameter will subsequently delete the activities from the Saved project. This is useful for creating an empty project with auxiliary files already attached. The deletion process is slow. A better alternative would be to open an empty template project and SaveAs.

SHOWLOG

Syntax: SHOWLOG

Description: Displays the OPBatch.log file.

TA | TIMEANALYSE | TIMEANALYZE

Syntax: TA [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8]

Description: This command performs a Time Analysis on the open project. All of the parameters are optional and, if they are not specified, any information stored currently in the Project Directory for a particular value is used (except for Auto Analysis, Parameter 3, as indicated below).

1 = Target Type

NONE

NE (not earlier than)

NL (not later than)

ON (on target)

2) If parameter 1 <> "None", the target finish date must be specified.

3) Analyze each time data changes ((True/False)--defaults to FALSE unless explicitly set, regardless of project setting).

4) Actual Date Option (True/False)

5) Show Summary Dates in Subprojects (True/False).

6) Multiple Ends (True/False).

7) Don't consider a subproject a finish activity if any child's successor's are outside of the subproject (True/False). Setting is honored only if parameter 6 (Multiple Ends) is true.

8) Out-of-sequence progress:

0 = Respect positive lags

1 = Ignore positive lags

2 = Ignore all predecessors

The value "USE_CURRENT" allows you to have a "placeholder" for a certain parameter without changing the Project Directory value associated with that parameter.

TIMENOW

Syntax: TIMENOW TimeNowDate

Description: This command sets the TimeNow date of the current project to the specified date. The date entered is interpreted using your current regional settings.

Open Plan Internal Batch Processor

When the Console Application (**OPPMultiInstanceBatch.exe**) creates an instance of Open Plan, it passes a series of options on the command line, which it creates from the information entered in the Console Batch file.

For the purposes of testing a Process Batch file (that is, a script), a single instance of OPP may be launched via the command line (like the Console Application). Invoking multiple instances of OPP in this manner may lead to errors as the Console Application prevents multiple instances from logging in simultaneously to the database.

The syntax for invoking OPP from the command line to execute a Process Batch file is as follows:

Syntax: OpenPlan.exe /NUserName /PPassword /BProcessBatchFile [/AProjectName]
[/XBatchSequenceNumber] [/SLogFileSuffix] /T

Note: There must not be a space between the forward slash + first character and the rest of the parameter.

- **/NUserName** – This parameter is mandatory. This is the Open Plan username. There is no space between the '/N' and the 'UserName'. For example, /NSysadmin.
- **/PPassword** – This parameter is mandatory. This is the Open Plan username's password. There is no space between the '/P' and the 'Password'. For example, /PPassword.
- **/BProcessBatchFile** – This parameter is mandatory and points to a valid, fully-qualified text file name (that is, Batch script). If there are any spaces in the file name, the fully-qualified path should be enclosed in quotation marks. For example, /B"C:\My Stuff\MyBatch.txt". There is no space between the '/B' and the first double quote of the fully-qualified text file path.
- **/AProjectName** – This parameter is optional. The Project argument should be the name of a project, but a value of NONE indicates that there is no default project for this instance. This means that the "OPEN @PROJ" syntax cannot be used within the Process Batch File (see comments about the OPEN command in the "Setting up The Process Batch File" section below).
- **/XBatchSequenceNumber** – This parameter is optional. This number must be in the range or 1 and 100,000. For purposes of internal batch mode, you may use any number within the range above. If you do not supply a log file suffix, that number could become the suffix for your log files.
- **/SLogFileSuffix** – This parameter is optional. The LogFileSuffix argument specifies an identifier that will be appended to the root name of an OPP log file so that log files created during the batch process for different batches may be differentiated from one another.

For example, if GMK is passed as a value, the opwin.log file would be named opwin_GMK.log. An underscore is added by Open Plan to separate the suffix from the root file name; however, including an underscore in the LogFileSuffix is permitted. The LogFileSuffix argument must be supplied, but if a value of DEFAULT is passed, Open Plan will form its suffix based on the starting sequence number (with a starting sequence number of 1000, the suffix generated for the first batch processed will be 1000; the second, 1001, and so on and so forth).

- **/T** – This parameter is mandatory. This parameter indicates to OPP that the process is not being called from the console app, and OPP avoids making certain checks that are necessary only for console mode. Technically, this parameter is optional, but if you are calling the Internal Batch, it should be considered mandatory.

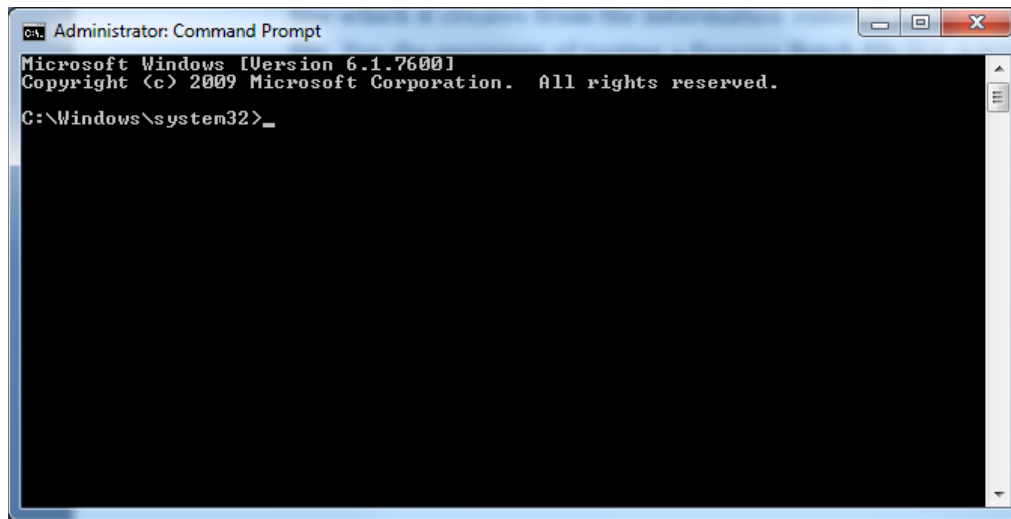
Example:

```
OpenPlan.EXE /Nuser1 /Ppw1 /B"C:\My Stuff\MyBatch.txt" /ACLEAN /X1 /SGMK
```

Note: The fourth through six arguments—ProjectName, BatchSequenceNumber, and LogFileSuffix—are optional. In addition, the batch file name in the example has been enclosed in quotation marks since there is a space in one of the folder names ("My Stuff").

Delttek recommends that you use the ADMINISTRATOR command prompt when launching the Console Application or Open Plan from the command line. This version of the command line may be invoked by right-clicking on the Command Prompt icon and selecting **Run as Administrator**.

The Command Prompt window has "Administrator: Command Prompt" as its title as shown below:



Open Plan API

The following automation calls return a long integer error result (and do not throw exceptions). OPP can use the error value to provide better error messages.

OPProject Object

```
TimeAnalyzeWithErrorResult (Boolean ShowDialog)
ResourceScheduleWithErrorResult(String TimeOrResourceLimited, Boolean
ShowDialog, Boolean DoTimeAnalysis)
CalculateCostWithErrorResult(Boolean ShowDialog, Boolean DoTimeAnalysis)
AutoProgressWithErrorResult(Boolean ShowDialog)
RiskAnalyzeWithErrorResult(Boolean ShowDialog)
UpdateBaselineWithErrorResult(String BaselineName, String Baseline Filter,
ShortInteger Flags)
GeneralExportWithErrorResult(String Spec, String FileName)
CreateBackupWithErrorResult(String FileName, Boolean Overwrite)
```

Note: The *IncludeBaselines* parameter is no longer used.

OPApplication Object

Methods

```
FileOpenWithErrorResult (String FileName, String FileType, String OpenAccessMode, Boolean
OpenAllSubprojects)
```

Note: FileType may be "PROJECT_NOVIEWS" which specifies that the startup view should not be opened.

```
DeleteProcessLogs (no argument, returns True if successful)
```

Properties

```
LogOutputCumulative (True/FALSE argument)
```

Note: Process Logs are not cleared; they are deleted and recreated. Therefore, calling a combination of DeleteProcessLogs followed by LogOutputCumulative = True will delete and recreate the logs.

About Deltek

Better software means better projects. Deltek delivers software and information solutions that enable superior levels of project intelligence, management and collaboration. Our industry-focused expertise makes your projects successful and helps you achieve performance that maximizes productivity and revenue. www.deltek.com