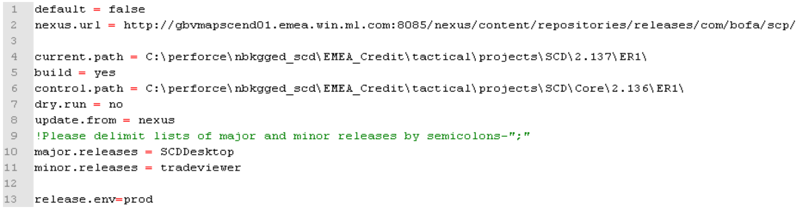
USE CASE NUMBER	USE CASE
1	Pre Production Build AND user wants to run version manager application

Step 1

Make sure the config.properties files looks like screenshot below. Depending on what the user wants the application will prompt if user wants to build (release environment other parameters MUST be included in config.properties file) or version

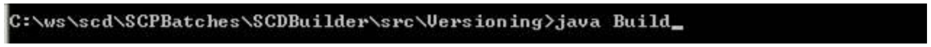


```
default = false
nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/

current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\
build = yes
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
dry.run = no
update.from = nexus
!Please delimit lists of major and minor releases by semicolons-";"
major.releases = SCDDesktop
minor.releases = tradeviewer

release.env=prod
```

Step 2

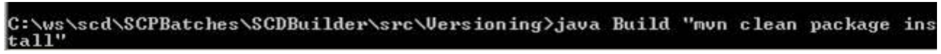
Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.

For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly (only updating portion)



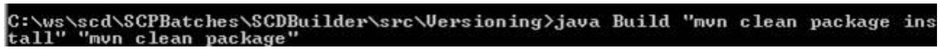
```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_
```

For example, 1 command arguments would resemble (see below). In addition, only the first portion of project is build (JAR files-SCTDesktop) which would help develop understand in JARs were updated correctly and built correctly



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall"
```

For example, 2 command arguments would resemble (see below). Entire project (JAR and WAR files are built and updated), developer would use this pre-production build as evidence the process works



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall" "mvn clean package"
```

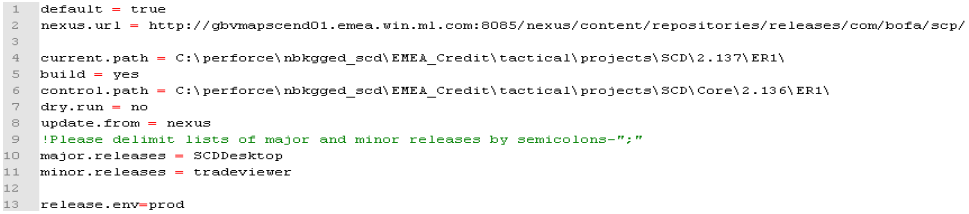
Step 3

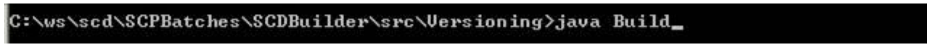
Application prompts user to "direct build" and user inputs YES. Thus, control file path, build environment and other version manager prompts are BYPASSED. Defaults that will determine version numbers are obtained prompts (Control File, Environment, Compare to nexus or control file etc...)

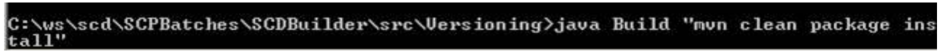
Expected Results

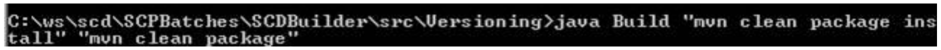
A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)

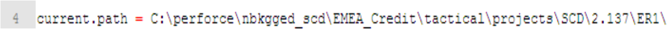
2	Pre Production Build but user wants to SKIP version manager application by including pertinent information (release environment, region, etc..) in the config.properties file of his/her project

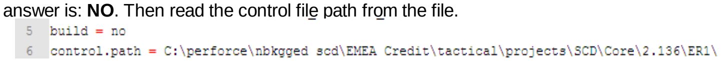
Step 1	Notice "default" is equal to meaning developing wants to SKIP version manager application prompts and version according to his/her configuraitons.property file (but note the developer DOES want to BUILD). Specify pertinent parameters in the config.properties file, for examples look at screenshot below:
	
	`default = true`
	`nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/`
	
	`current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\`
	`build = yes`
	`control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\`
	`dry.run = no`
	`update.from = nexus`
	`!Please delimit lists of major and minor releases by semicolons-";"`
	`major.releases = SCDDesktop`
	`minor.releases = tradeviewer`
	
	`release.env=prod`

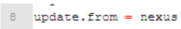
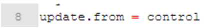
Step 2	Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.
	For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly (only updating portion)  `C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_`

	For example, 1 command arguments would resemble (see below). In addition, only the first portion of project is build (JAR files-SCTDesktop) which would help develop understand in JARs were updated correctly and built correctly
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall"`

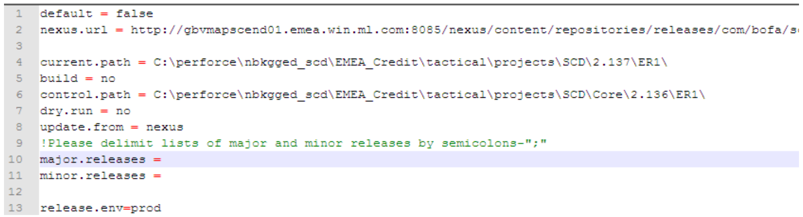
	For example, 2 command arguments would resemble (see below). Entire project (JAR and WAR files are built and updated), developer would use this pre-production build as evidence the process works
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall" "mvn clean package"`

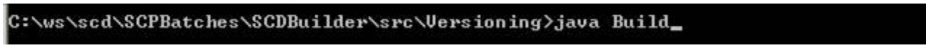
Step 3	Application reads current project source path from the config.properties file shown above. In this example, the path is:  `current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\`

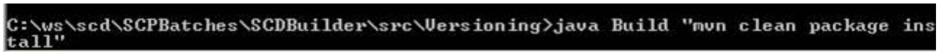
Step 4	Application reads from the config.properties file shown above whether user to directly build the project (yes) or update versioning (no), in this example the answer is: **NO**. Then read the control file path from the file.
	
	`build = no`
	`control.path = C:\perforce\nbkgged scd\EMEA Credit\tactical\projects\SCD\Core\2.136\ER1\`

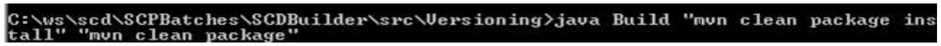
Step 5	Application reads from the config.properties file shown above whether user wants to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps) For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .
	If indicate **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6.  `update.from = nexus`
	If indicate **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.  `update.from = control`
Step 6	In config.properties file, since "major.releases" and "minor.releases" have no values specified, the update type is "default release", which increments the patch version number only. In the above example, IF there's difference, version manager will update the version to 1.0.7 if "NEXUS" is chosen in Step 5 OR 1.0.5 if "CONTROL" is chosen in Step 5.
Step 7	Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)
Expected Results	A pre production run we only expect to see the command line outputs based on a USER modified configuration.properties file

3 Pre production build and user wants to run version manager application. In addition, user wants to version according to patch release

Step 1	Make sure the config.properties files looks like screenshot below.
	
	`default = false`
	`nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/s`
	
	`current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\`
	`build = no`
	`control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\`
	`dry.run = no`
	`update.from = nexus`
	`!Please delimit lists of major and minor releases by semicolons-";"`
	`major.releases =`
	`minor.releases =`
	
	`release.env=prod`

Step 2	Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.
	For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.  `C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_`

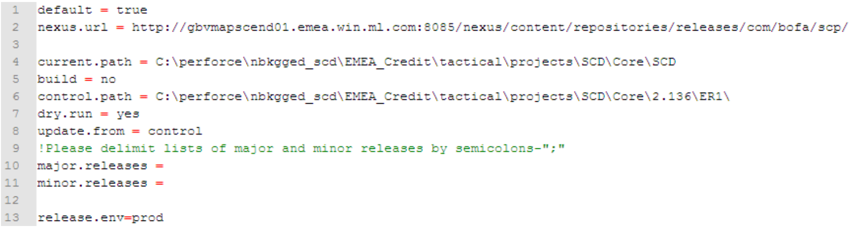
	For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall"`

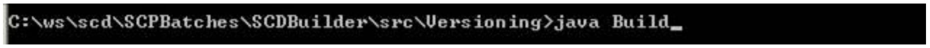
	For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall" "mvn clean package"`

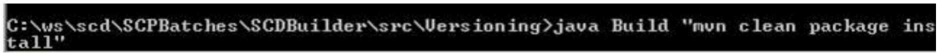
Step 3	Application prompts user for the current project source path, for example, user can input: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD\
Step 4	Application prompts user to directly build the project (yes) or update versioning (no), user input: **NO**. Then the application prompts user for the control file path (the released version to be compared with), for example: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
Step 5	Application prompts user to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps) For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .
	User input **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6
	User input **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.
Step 6	In config.properties file, since "major.releases" and "minor.releases" have no values specified, the update type is "default release", which increments the patch version number only. In the above example, IF there's difference, version manager will update the version to 1.0.7 if "NEXUS" is chosen in Step 5 OR 1.0.5 if "CONTROL" is chosen in Step 5. NOTE: the version number looks like 1 . 0 . 6 major-release-number.minor-release-number.patch-release-number

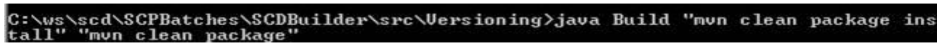
Step 7	Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: **YES**.
Expected Results	Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)

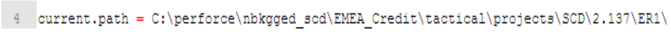
4 Pre production build and user wants to BYPASS version manager application (user must specify pertinent information such as environment). In addition, user wants to version according to patch release

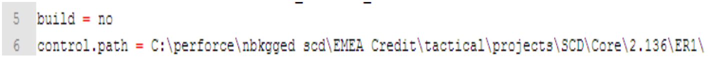
Step 1	Make sure the config.properties files looks like screenshot below.
	
	`default = true`
	`nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/`
	
	`current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD`
	`build = no`
	`control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\`
	`dry.run = yes`
	`update.from = control`
	`!Please delimit lists of major and minor releases by semicolons-";"`
	`major.releases =`
	`minor.releases =`
	
	`release.env=prod`

Step 2	Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.
	For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.  `C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_`

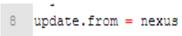
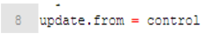
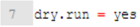
	For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall"`

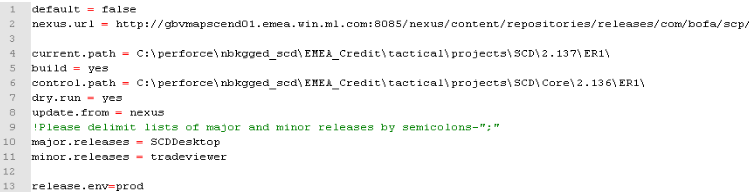
	For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.
	
	`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
	`tall" "mvn clean package"`

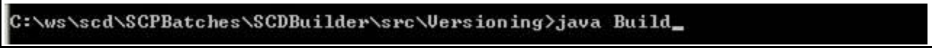
Step 3	Application reads current project source path from the config.properties file shown above. In this example, the path is:  `current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\`

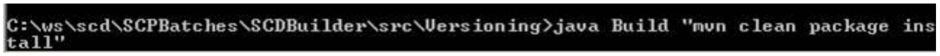
Step 4	Application reads from the config.properties file shown above whether user to directly build the project (yes) or update versioning (no), in this example the answer is: **NO**. Then read the control file path from the file.
	
	`build = no`
	`control.path = C:\perforce\nbkgged scd\EMEA Credit\tactical\projects\SCD\Core\2.136\ER1\`

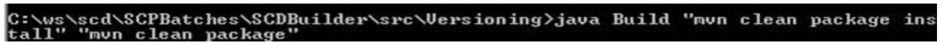
	Application reads from the config.properties file shown above whether user wants to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps) For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .

		Step 5	If indicate **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6.  `update.from = nexus`
			If indicate **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.  `update.from = control`
		Step 6	In config.properties file, since "major.releases" and "minor.releases" have no values specified, the update type is "default release", which increments the patch version number only. In the above example, IF there's difference, version manager will update the version to 1.0.7 if "NEXUS" is chosen in Step 5 OR 1.0.5 if "CONTROL" is chosen in Step 5.
		Step 7	Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: **YES**.  `dry.run = yes`
		Expected Results	Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)
5	Pre production build in which the user runs the version manager application. In addition the user also specifies that versioning must be done to either or major and minor releases		

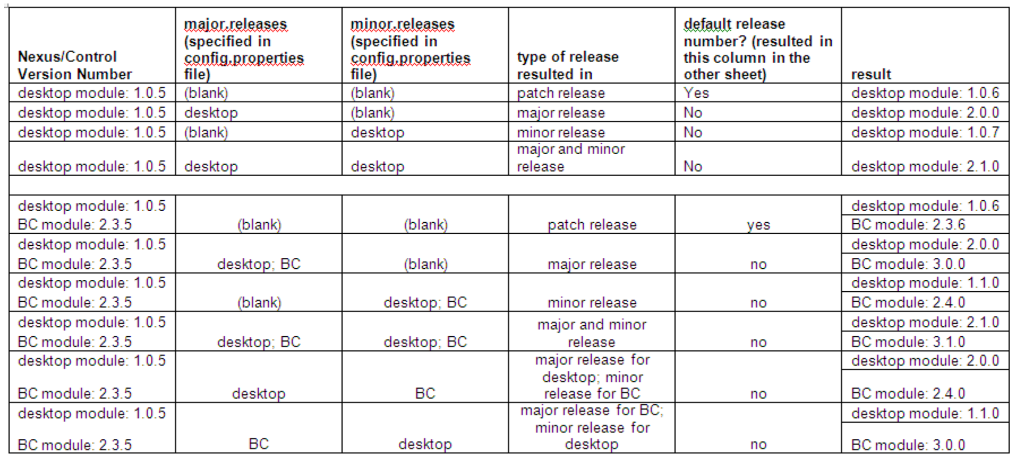
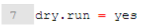
		Step 1	Make sure the config.properties files looks like screenshot below.
			
			`default = false`
			`nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/`
			
			`current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\`
			`build = yes`
			`control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\`
			`dry.run = yes`
			`update.from = nexus`
			`!Please delimit lists of major and minor releases by semicolons-";"`
			`major.releases = SCDDesktop`
			`minor.releases = tradeviewer`
			
			`release.env=prod`

		Step 2	Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the
			For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.  `C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_`

			For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.
			
			`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
			`tall"`

			For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.
			
			`C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins`
			`tall" "mvn clean package"`

		Step 3	Application prompts user for the current project source path, for example, user can input: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD\
		Step 4	Application prompts user to directly build the project (yes) or update versioning (no), user input: **NO**. Then the application prompts user for the control file path (the released version to be compared with), for example: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\

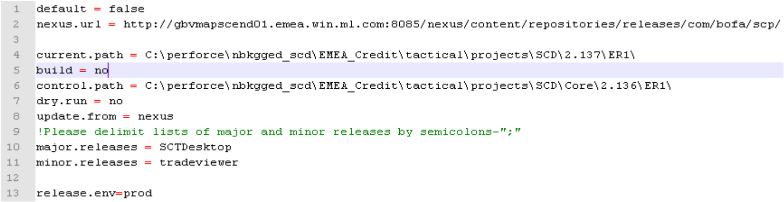
<table>
<tr><td rowspan="3">Step 5</td><td>Application prompts user to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps)
For example, if the latest version in Nexus for desktop module is 1.1.6 AND the input control file version for desktop module is 1.2.4.</td></tr>
<tr><td>User input <b>NEXUS</b>, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.1.6</td></tr>
<tr><td>User input <b>CONTROL</b>, then the application will compare the desktop module with the nexus latest release and update the version number based on 2.2.4.</td></tr>
<tr><td>Step 6</td><td>See versioning matrix below for examples of incrementing major/minor and patch releases

<table>
<tr><th>Nexus/Control Version Number</th><th>major.releases (specified in config.properties file)</th><th>minor.releases (specified in config.properties file)</th><th>type of release resulted in</th><th>default release number? (resulted in this column in the other sheet)</th><th>result</th></tr>
<tr><td>desktop module: 1.0.5</td><td>(blank)</td><td>(blank)</td><td>patch release</td><td>Yes</td><td>desktop module: 1.0.6</td></tr>
<tr><td>desktop module: 1.0.5</td><td>desktop</td><td>(blank)</td><td>major release</td><td>No</td><td>desktop module: 2.0.0</td></tr>
<tr><td>desktop module: 1.0.5</td><td>(blank)</td><td>desktop</td><td>minor release</td><td>No</td><td>desktop module: 1.0.7</td></tr>
<tr><td>desktop module: 1.0.5</td><td>desktop</td><td>desktop</td><td>major and minor release</td><td>No</td><td>desktop module: 2.1.0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>(blank)</td><td>(blank)</td><td>patch release</td><td>yes</td><td>desktop module: 1.0.6
BC module: 2.3.6</td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>desktop; BC</td><td>(blank)</td><td>major release</td><td>no</td><td>desktop module: 2.0.0
BC module: 3.0.0</td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>(blank)</td><td>desktop; BC</td><td>minor release</td><td>no</td><td>desktop module: 1.1.0
BC module: 2.4.0</td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>desktop; BC</td><td>desktop; BC</td><td>major and minor release</td><td>no</td><td>desktop module: 2.1.0
BC module: 3.1.0</td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>desktop</td><td>BC</td><td>major release for desktop; minor release for BC</td><td>no</td><td>desktop module: 2.0.0
BC module: 2.4.0</td></tr>
<tr><td>desktop module: 1.0.5
BC module: 2.3.5</td><td>BC</td><td>desktop</td><td>major release for BC; minor release for desktop</td><td>no</td><td>desktop module: 1.1.0
BC module: 3.0.0</td></tr>
</table></td></tr>
<tr><td>Step 7</td><td>Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: <b>YES</b>.

dry.run = yes</td></tr>
<tr><td>Expected Results</td><td>Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)</td></tr>
</table>

6 Pre production build in which the user BYPASS the version manager application (and wants to version according to parameters specified in the config.properties file). In addition the user also specifies that versioning must be done to either or major and minor releases

7 User builds specifying an environment. User also executes the version manager application. In addition the user also specifies that versioning must be done to either or major and minor releases

Step 1

Make sure the config.properties files looks like screenshot below.

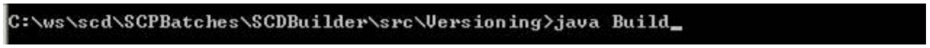


```
default = false
nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/

current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\
build = no
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
dry.run = no
update.from = nexus
!Please delimit lists of major and minor releases by semicolons-";"
major.releases = SCTDesktop
minor.releases = tradeviewer

release.env=prod
```

Step 2

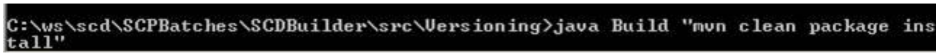
Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.

For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.



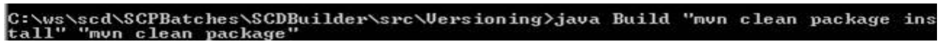
```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_
```

For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall"
```

For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall" "mvn clean package"
```

Step 3

Application prompts user for the current project source path, for example, user can input:
C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD\

Step 4

Application prompts user to directly build the project (yes) or update versioning (no), user input: **NO**. Then the application prompts user for the control file path (the released version to be compared with), for example: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\

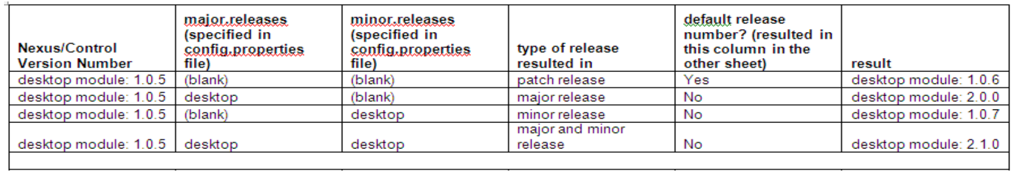
Step 5

Application prompts user to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps)
For example, if the latest version in Nexus for desktop module is 1.1.6 AND the input control file version for desktop module is 1.2.4.

User input **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.1.6

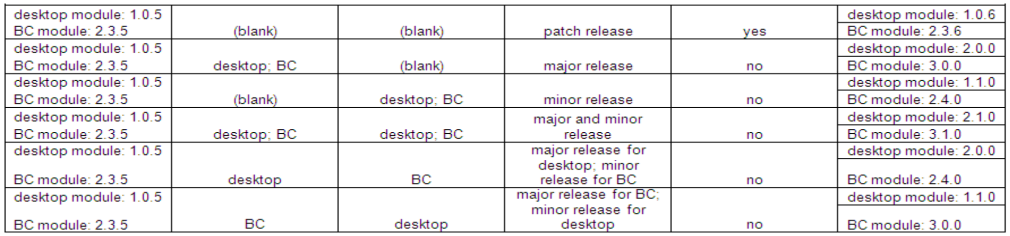
User input **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 2.2.4.

See versioning matrix below for examples of incrementing major/minor and patch releases



Nexus/Control Version Number	major.releases (specified in config.properties file)	minor.releases (specified in config.properties file)	type of release resulted in	default release number? (resulted in this column in the other sheet)	result
desktop module: 1.0.5	(blank)	(blank)	patch release	Yes	desktop module: 1.0.6
desktop module: 1.0.5	desktop	(blank)	major release	No	desktop module: 2.0.0
desktop module: 1.0.5	(blank)	desktop	minor release	No	desktop module: 1.0.7
desktop module: 1.0.5	desktop	desktop	major and minor release	No	desktop module: 2.1.0

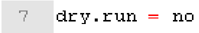
Step 6



desktop module: 1.0.5 BC module: 2.3.5	(blank)	(blank)	patch release	yes	desktop module: 1.0.6 BC module: 2.3.6
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	(blank)	major release	no	desktop module: 2.0.0 BC module: 3.0.0
desktop module: 1.0.5 BC module: 2.3.5	(blank)	desktop; BC	minor release	no	desktop module: 1.1.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	desktop; BC	major and minor release	no	desktop module: 2.1.0 BC module: 3.1.0
desktop module: 1.0.5 BC module: 2.3.5	desktop	BC	major release for desktop; minor release for BC	no	desktop module: 2.0.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	BC	desktop	major release for BC; minor release for desktop	no	desktop module: 1.1.0 BC module: 3.0.0

Step 7

Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: **YES**.



```
dry.run = no
```

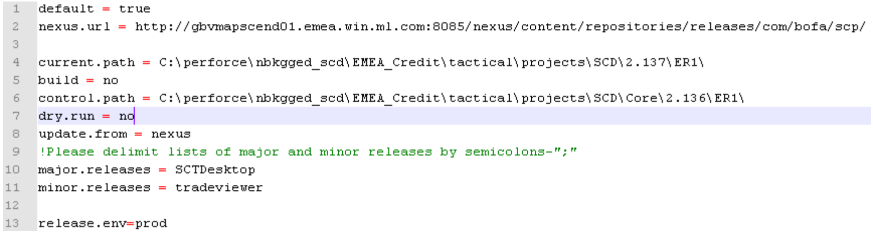
Expected Results

Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only

8 User builds specifying an environment. BYPASS the version manager application (and wants to version according to parameters specified in the config.properties file). In addition the user also specifies that versioning must be done to either or major and minor releases

Step 1

Make sure the config.properties files looks like screenshot below.

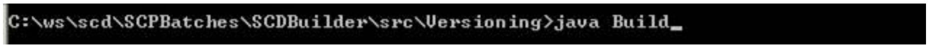


```
default = true
nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/

current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\
build = no
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
dry.run = no
update.from = nexus
!Please delimit lists of major and minor releases by semicolons-";"
major.releases = SCTDesktop
minor.releases = tradeviewer

release.env=prod
```

Step 2

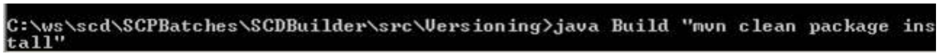
Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.

For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.



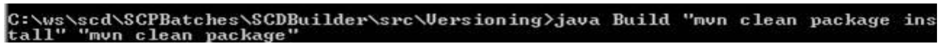
```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_
```

For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall"
```

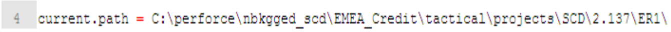
For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall" "mvn clean package"
```

Step 3

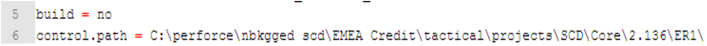
Application reads current project source path from the config.properties file shown above. In this example, the path is:



```
current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\2.137\ER1\
```

Step 4

Application reads from the config.properties file shown above whether user to directly build the project (yes) or update versioning (no), in this example the answer is: **NO**. Then read the control file path from the file.

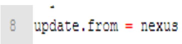


```
build = no
control.path = C:\perforce\nbkgged scd\EMEA Credit\tactical\projects\SCD\Core\2.136\ER1\
```

Step 5

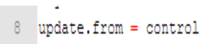
Application reads from the config.properties file shown above whether user wants to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps)
For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .

If indicate **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6.



```
update.from = nexus
```

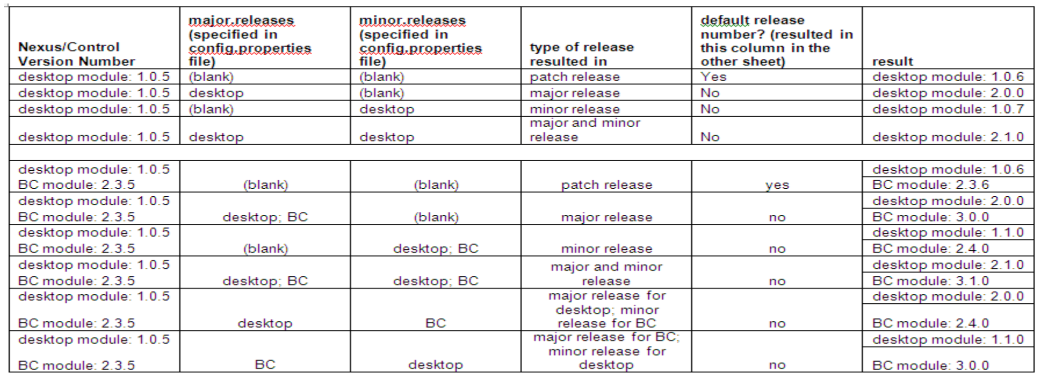
If indicate **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.



```
update.from = control
```

Step 6

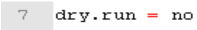
See versioning matrix below for examples of incrementing major/minor and patch releases



Nexus/Control Version Number	major.releases (specified in config.properties file)	minor.releases (specified in config.properties file)	type of release resulted in	default release number? (resulted in this column in the other sheet)	result
desktop module: 1.0.5	(blank)	(blank)	patch release	Yes	desktop module: 1.0.6
desktop module: 1.0.5	desktop	(blank)	major release	No	desktop module: 2.0.0
desktop module: 1.0.5	(blank)	desktop	minor release	No	desktop module: 1.0.7
desktop module: 1.0.5	desktop	desktop	major and minor release	No	desktop module: 2.1.0
desktop module: 1.0.5 BC module: 2.3.5	(blank)	(blank)	patch release	yes	desktop module: 1.0.6 BC module: 2.3.6
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	(blank)	major release	no	desktop module: 2.0.0 BC module: 3.0.0
desktop module: 1.0.5 BC module: 2.3.5	(blank)	desktop; BC	minor release	no	desktop module: 1.1.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	desktop; BC	major and minor release	no	desktop module: 2.1.0 BC module: 3.1.0
desktop module: 1.0.5 BC module: 2.3.5	desktop	BC	major release for desktop; minor release for BC	no	desktop module: 2.0.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	BC	desktop	major release for BC; minor release for desktop	no	desktop module: 1.1.0 BC module: 3.0.0

Step 7

Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: **YES**.

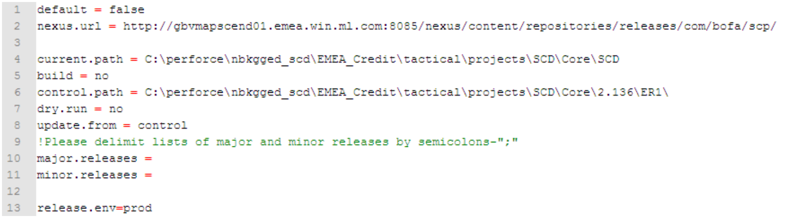


```
dry.run = no
```

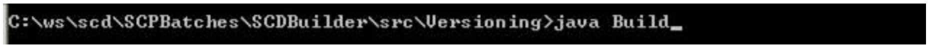
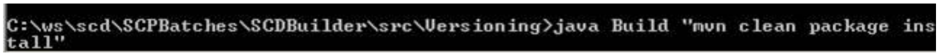
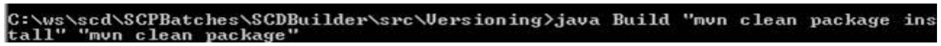
Expected Results

Updated version number will show in the newly generated pom.xml.test files but the original pom.xml will stay as not modified. A pre production run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)

9 User wants to execute version manager appliccation and build with specified environment (prod, QA, str, etc). User also wants to version according to patch release number

<table>
<tr><td>Step 1</td><td>Make sure the config.properties files looks like screenshot below.

<pre>
default = false
nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/

current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD
build = no
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
dry.run = no
update.from = control
!Please delimit lists of major and minor releases by semicolons-";"
major.releases =
minor.releases =

release.env=prod
</pre></td></tr>
<tr><td rowspan="4">Step 2</td><td>Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.</td></tr>
<tr><td>For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.

<pre>
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_
</pre></td></tr>
<tr><td>For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.

<pre>
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall"
</pre></td></tr>
<tr><td>For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.

<pre>
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall" "mvn clean package"
</pre></td></tr>
<tr><td>Step 3</td><td>Application prompts user for the current project source path, for example, user can input:
C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD\</td></tr>
<tr><td>Step 4</td><td>Application prompts user to directly build the project (yes) or update versioning (no), user input: <b>NO</b>. Then the application prompts user for the control file path (the released version to be compared with), for example: C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\</td></tr>
<tr><td rowspan="3">Step 5</td><td>Application prompts user to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps)
For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .</td></tr>
<tr><td>User input <b>NEXUS</b>, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6</td></tr>
<tr><td>User input <b>CONTROL</b>, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.</td></tr>
<tr><td>Step 6</td><td>In config.properties file, since "major.releases" and "minor.releases" have no values specified, the update type is "default release", which increments the patch version number only. In the above example, IF there's difference, version manager will update the version to 1.0.7 if "NEXUS" is chosen in Step 5 OR 1.0.5 if "CONTROL" is chosen in Step 5.

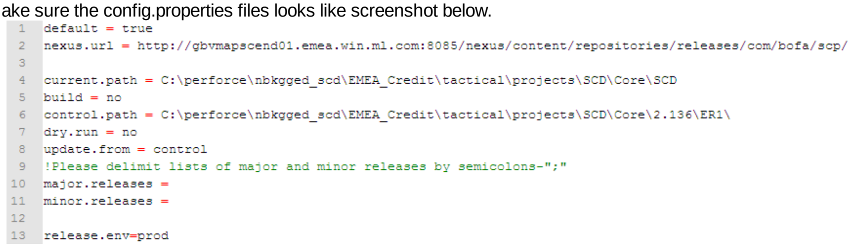
NOTE: the version number looks like
1 . 0 . 6
major-release-number.minor-release-number.patch-release-number</td></tr>
<tr><td>Step 7</td><td>Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: <b>NO</b> to overwrite the original pom.xml files .</td></tr>
</table>

Expected Results	Updated version number will show in the original pom.xml files (they are overwritten). An actual build run we only expect to see the command line outputs (indivudual results for command lines are explained in Step 2)

10 User wants to BYPASS version manager appliccation and build with specified (in config.properties file) environment (prod, QA, str, etc). User also wants to version according to patch release number

Step 1

Make sure the config.properties files looks like screenshot below.

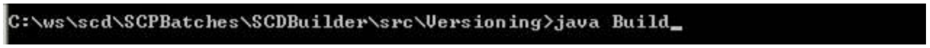


```
default = true
nexus.url = http://gbvmapscend01.emea.win.ml.com:8085/nexus/content/repositories/releases/com/bofa/scp/

current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD
build = no
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
dry.run = no
update.from = control
!Please delimit lists of major and minor releases by semicolons-";"
major.releases =
minor.releases =

release.env=prod
```

Step 2

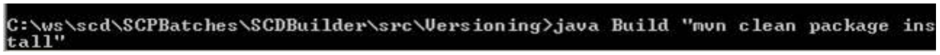
Run the version manager (java application). See command line examples below . In the command line the number of arguments specified determines the end result.

For example 0 command arguments would resemble (see below). In addition a developer would not include any arguments in command to verify updating of pom.xmls was completed correctly. The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files are updated. Since no command line argument is specified, neither SCTDesktop nor webstart project is built. In all, the application does nothing to the SCD application.



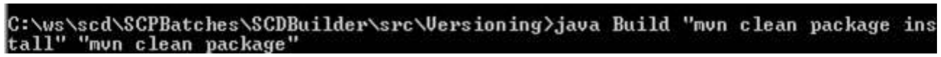
```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build_
```

For example 1 command arguments would resemble (see below).The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Since one command line argument is specified, the application builds the SCTDesktop project based on the command line argument but not the webstart project. In all, the WAR file is not built.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall"
```

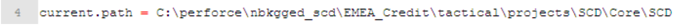
For example 2 command arguments would resemble (see below). The build uses conditions specified in the config.properties file. Since direct build is specified as "no", the pom.xml files will be updated. Two (or more) command line arguments are specified, SCTDesktop project is built based on the first command line argument and webstart project is built based on the second command line argument. Finally the built WAR file is renamed to be "SCTDesktop-env" where env is the value specified in the "release.env" field in config.properties file.



```
C:\ws\scd\SCPBatches\SCDBuilder\src\Versioning>java Build "mvn clean package ins
tall" "mvn clean package"
```

Step 3

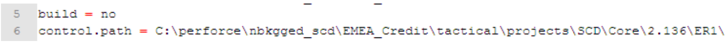
Application reads current project source path from the config.properties file shown above. In this example, the path is:



```
current.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\SCD
```

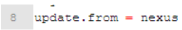
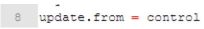
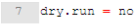
Step 4

Application reads from the config.properties file shown above whether user to directly build the project (yes) or update versioning (no), in this example the answer is: **NO**. Then read the control file path from the file.



```
build = no
control.path = C:\perforce\nbkgged_scd\EMEA_Credit\tactical\projects\SCD\Core\2.136\ER1\
```

Application reads from the config.properties file shown above whether user wants to use Nexus's latest release or the control file as the baseline to update the pom file. (Different answer specified in the following steps)
For example, if the latest version in Nexus for desktop module is 1.0.6 AND the input control file version for desktop module is 1.0.4. .

Step 5	If indicate **NEXUS**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.6.  `update.from = nexus`
	If indicate **CONTROL**, then the application will compare the desktop module with the nexus latest release and update the version number based on 1.0.4.  `update.from = control`
Step 6	In config.properties file, since "major.releases" and "minor.releases" have no values specified, the update type is "default release", which increments the patch version number only. In the above example, IF there's difference, version manager will update the version to 1.0.7 if "NEXUS" is chosen in Step 5 OR 1.0.5 if "CONTROL" is chosen in Step 5.
Step 7	Application prompts user if this is a dry-run or not (dry-run creates new pom.xml.test files instead of overwriting the original pom.xml file). User input: NO to overwrite the original pom.xml files .  `dry.run = no`
Expected Results	Updated version number will show in the original pom.xml files (they are overwritten). An actual build run we only expect to see the command line outputs

Versioning Matrix

Nexus/Control Version Number	**major.releases (specified in config.properties file)**	**minor.releases (specified in config.properties file)**	**type of release resulted in**	**default release number? (resulted in this column in the other sheet)**	**result**
desktop module: 1.0.5	(blank)	(blank)	patch release	Yes	desktop module: 1.0.6
desktop module: 1.0.5	desktop	(blank)	major release	No	desktop module: 2.0.0
desktop module: 1.0.5	(blank)	desktop	minor release	No	desktop module: 1.0.7
desktop module: 1.0.5	desktop	desktop	major and minor release	No	desktop module: 2.1.0
desktop module: 1.0.5 BC module: 2.3.5	(blank)	(blank)	patch release	yes	desktop module: 1.0.6 BC module: 2.3.6
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	(blank)	major release	no	desktop module: 2.0.0 BC module: 3.0.0
desktop module: 1.0.5 BC module: 2.3.5	(blank)	desktop; BC	minor release	no	desktop module: 1.1.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	desktop; BC	desktop; BC	major and minor release	no	desktop module: 2.1.0 BC module: 3.1.0
desktop module: 1.0.5 BC module: 2.3.5	desktop	BC	major release for desktop; minor release	no	desktop module: 2.0.0 BC module: 2.4.0
desktop module: 1.0.5 BC module: 2.3.5	BC	desktop	major release for BC; minor release for desktop	no	desktop module: 1.1.0 BC module: 3.0.0

Note For multiple modules that are specified in the major.release or minor.release key field in config.properties file, separate them by ";"

Modules can be specified in the major.release and minor.release fields:

desktop
Ivol
BaseCorrelation
BasketManager
reporting framework
reportviewer
RFL
SCK
tradeviewer
SCTDesktop
ConfigurationsUat
ConfigurationsProd
ConfigurationsUatDualServer
ConfigurationsStr

File Compare If two directories contain the same files, then do not update the module's version number in the module's pom file. Otherwise, increase version number based on the requirement shown in "Version Matrix" tag

Use Case #	Use Case Description	Directory 1 Content	Directory 2 Content	Directory Compare Result	Update version number?	Comment
1	Same **amount** of files in both directory and file **names** and **content** are the same	abc.doc	abc.doc	Same	No	File order does not matter.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
2	Same **amount** of files in both directory and file **names** are the same but **content** is difference. Difference exists in non-picture type files.	abc.doc def.txt	abc.doc (content different from abc.doc in directory 1 def.txt	Different	Yes	File order does not matter.
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
3	Same **amount** of files in both directory and file **names** are the same but **content** is difference. Difference exists in picture type files.	abc.doc	abc.doc	Same	No	Currently the file comparator cannot detect the difference in pic, jpg, gif (picture types) files.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic (picture content is different from picture.pic in directory 1)			
		graph.jpg	graph.jpg			
4	Same amount of files in both direcotry. Two files have different names but the same content.	abc.doc	abcdef.doc (content same as abc.doc in directory 1)	Different	Yes	File order does not matter.
		def.txt config.properties	def.txt config.properties			
		help.java	help.java			

		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
5	Same **amount** of files in both directory and file **names** and **content** both different	abc.doc	abcdef.doc (content different from abc.doc in directory 1)	Different	Yes	File order does not matter.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
		abc.doc	abc.jar	Different	Yes	File order does not matter.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
6	Same **amount** of files in both directory and file **names** and **content** both different	abc.doc	abc.doc (content	Different	Yes	File order does not matter.
		def.txt	abcdef.txt (content same as def.txt in directory 1)			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
	Different **amount** of files in both directory and file **names** and **content** are the same (Directory	abc.doc	abc.doc			
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			

7	**content** are the same (Directory 1 has more files)	file.jsp	file.jsp	Different	Yes	File order does not matter. File type does not matter.
		picture.pic	picture.pic			
		graph.jpg				
	Different **amount** of files in both directory and file **names** and **content** are the same (Directory 2 has more files)	abc.doc	abc.doc			
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
			test.xsl			
8	Different **amount** of files in both directory and file **names** are the same but **content** is difference. Difference exists in non-picture type files. (Directory 1 has more files)	abc.doc	abc.doc (content different from abc.doc in directory 1)	Different	Yes	File order does not matter. File type does not matter.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic graph.jpg	picture.pic			
	Different **amount** of files in both directory and file **names** and **content** are the same (Directory 2 has more files)	abc.doc def.txt config.properties help.java file.jsp picture.pic graph.jpg	abc.doc (content different from abc.doc in directory 1) def.txt config.properties help.java file.jsp picture.pic graph.jpg test.xsl			
	Different **amount** of files in both directory and file **names** are the same but **content** is difference	abc.doc def.txt config.properties help.java file.jsp	abc.doc def.txt config.properties file.jsp			

9	same but **content** is difference. Difference exists in picture type files. (Directory 1 has more files)	picture.pic graph.jpg	picture.pic (picture content is different from picture.pic in directory 1)	Different	Yes	File order does not matter. File type does not matter.
	Different **amount** of files in both directory and file **names** and **content** are the same (Directory 2 has more files)	abc.doc def.txt	abc.doc def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic (picture content is different from picture.pic in directory 1)			
		graph.jpg	graph.jpg			
			test.xsl			
10	Different amount of files in both direcotry. Two files have different names but the same content. (Directory 1 has more files)	abc.doc	abcdef.doc (content same as abc.doc in directory 1)	Different	Yes	File order does not matter. File type does not matter.
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg				
	Different **amount** of files in both directory and file **names** and **content** are the same (Directory 2 has more files)	abc.doc	abcdef.doc (content same as abc.doc in directory 1)			
		def.txt	def.txt			
		config.properties help.java	config.properties help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg	graph.jpg			
			test.xsl			
	Different **amount** of files in both directory and file **names** and	abc.doc	abcdef.doc (content different from abc.doc in directory 1)			
		def.txt	def.txt			

11	directory and file **names** and **content** both different. (Directory 1 has more files)	config.properties	config.properties	Different	Yes	File order does not matter. File type does not matter.
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg				
	Different **amount** of files in both directory and file **names** and **content** are the same. (Directory 2 has more files)	abc.doc	abcdef.doc (content different from abc.doc in directory 1)			
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp picture.pic	file.jsp picture.pic			
		graph.jpg	graph.jpg			
			test.xsl			
	Different **amount** of files in both directory and file **names** and **content** both different. (Directory 1 has more files)	abc.doc	abc.jar			
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp	file.jsp			
		picture.pic	picture.pic			
		graph.jpg				
	Different **amount** of files in both directory and file **names** and **content** are the same. (Directory 2 has more files)	abc.doc	abc.jar			
		def.txt	def.txt			
		config.properties	config.properties			
		help.java	help.java			
		file.jsp picture.pic	file.jsp picture.pic			
		graph.jpg	graph.jpg			
			test.xsl			
	Different **amount** of files in both directory and file **names** and **content** both different	abc.doc	abc.doc (content different from abc.doc in directory 1)			
		def.txt	abcdef.txt (content same as def.txt in directory 1)			
		config.properties	config.properties			
		help.java	help.java			

12		file.jsp	file.jsp	Different	Yes	File order does not matter. File type does not matter.
		picture.pic	picture.pic			
		graph.jpg				
	Different **amount** of files in both directory and file **names** and **content** are the same	abc.doc	abc.doc (content different from abc.doc in directory 1)			
		def.txt	abcdef.txt (content same as def.txt in directory 1)			
		config.properties	config.properties			
		help.java	help.java			