P4Convert: User Guide

Oct 2014

P4Convert: User Guide
Oct 2014

Copyright © 2007-2014 Perforce Software.

All rights reserved.

P4Convert is supplied by Perforce Software in the hope that it will be useful. It is a support utility, not a Perforce product. All use of this software is at the user's own risk and subject to the following terms and conditions.

THIS SOFTWARE IS PROVIDED BY PERFORCE SOFTWARE, INC. 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL PERFORCE SOFTWARE, INC. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

What is p4convert for? v

Import Mode (front door) v
Convert Mode (back door) v
Incremental Updates (front door) v

Chapter 1 Setup 1

System Requirements 1
Resource Tips: 1
Caveats 1
Setup and Usage 2
Generating a Subversion Dump file 2
Selecting a CVS root 3

Chapter 2 Configuration 5

Core converter settings. 5
General Perforce converter settings. 5
General Subversion converter settings. 6
General CVS converter settings. 6
Subversion: Selective and Incremental Conversions 7
Filtering Subversion paths 7
Labeling Subversion Paths 8
Changelist Offset options 10
Unicode Support 10
Normalisation 11
Subversion Properties 11
Advanced Configuration 12
Directory Properties 12
Empty changelists 12
Username translation 12
Binary file detection 12
Changelist Description Format 13
Case Sensitivity 13
RCS Keyword expansion (svn:keywords) 14
Merge Information (svn:mergeinfo) 14

Chapter 3 Running P4Convert 15

Import Mode 15
Configuration options specific to Import Mode 15
Convert Mode 16
Configuration options specific to Conversion Mode 16
Post Conversion [Conversion Mode] 17

Chapter 4 Notes 19

Keyword Expansion Issues: 19
Verification 19
Output and Logs 19
Console output and logging configuration options 19
These options are reserved for testing or future enhancements: 20
Reading Console/Logging output 20
Errors 22

What is p4convert for?

The p4convert conversion tool imports data from Subversion or CVS and reconstructs the file revision history in Perforce. For CVS the data is read from the CVSROOT and for Subversion data is read from a dump file. Data is added to Perforce in one of two ways:

Import Mode (front door)

Revisions are imported into a running Perforce Server. The Perforce Server could be new or contain existing data, but it must be running. Subversion revisions are added sequentially; file content is 'imported' and metadata such as file attributes, descriptions, dates and usernames are added to the Perforce changelist. CVS data is sorted into changeslists, based on available metadata such as date/time, author, change descriptions and file actions.

The recommended use of Import Mode is to start a new Perforce server to import each set of changes into a Perforce changelist. Using this method will produce a one-to-one mapping of Subversion revision numbers to Perforce changelists and a reproducible set of CVS changes. The new Perforce server can be used in isolation to confirm the success of the migration and then merged with an existing Perforce Server using the PerfMerge++ tool.

Convert Mode (back door)

Revisions are converted in full, creating a Perforce journal and set of archive files. Once replayed, the resulting conversion is just as if the changes had always been in Perforce. The results can be merged into an existing Perforce Server using the PerfMerge++ tool. The Import Mode is considered to be the safest method as all files are imported through a Perforce Client. Convert Mode is significantly faster, but requires an understanding of Perforce database administration.

Convert Mode is an advanced feature and should only be attempted by a Perforce expert familiar with replaying and upgrading databases. Incremental conversions are not supported using Convert Mode; however Import Mode can be used after a Convert Mode conversion to update the migration.

Incremental Updates (front door)

Incremental Subversion updates are possible only using the Import Mode where new revisions from Subversion are imported into Perforce. However, no changes should be made to the import area of the Perforce Server or conflicts may occur.

Incremental CVS imports are not currently supported in either Import or Convert Mode.

Chapter 1

Setup

System Requirements

1. [SVN] an unzipped Subversion dump file of the repository to be converted (generated without the *--delta* flag).
2. [CVS] an unzipped CVSROOT directory containing the RCS files of the repository to be converted.
3. System must have **Java SE Runtime 1.7**.
4. System must have the Perforce Server binaries 2010.2 or greater.
5. [Import Mode] A running Perforce Server (typically empty) with no pending changes in the conversion area.
6. [Convert Mode] An empty Perforce Root directory for the generated archive and journal files.

Resource Tips:

Running some conversions, particularly in Conversion Mode can be very memory intensive, as a mapping of revisions and integrations is kept in memory. For large conversions, or conversions with excessive branching, more memory will needed by the JVM.

The *--info* option will scan the Subversion dump file and report usage data to help estimate the required memory for Convert Mode. For example:

```
java -jar p4convert.jar --type=SVN --repo=<path_to_svn_dumpfile> --info
```

Use the *-Xmx* flags to increase the memory allocated to the JVM. For example, to allocate 64GB (65536MB) use:

```
java -Xmx65536M -jar p4convert.jar --config=myConfig.cfg
```

Caveats

1. The following table lists the combinations of tested operating systems and Perforce Server versions.

Operating System	10.2	14.1
Ubuntu 10.04	supported	supported
Darwin 11.4.0	supported	supported
Windows 7x64	supported	supported
SunOS 5.10	supported	supported

It is likely that other combinations will work, however it is not possible to test all combinations.

Please contact Perforce if you have any OS or version queries.

2. [Import Mode] Symbolic links are not supported in Import Mode on Windows.
3. Perforce keyword expansions are different from the Subversion standard and may yield different results when synced. CVS keyword detection is not yet supported.

Setup and Usage

Download the self-contained JAR from Perforce and check that Java SE runtime is installed with version 1.7 or greater.

For usage details, execute the jar with no options:

```
usage: java -jar p4convert.jar
 -c,--config <arg>    Use configuration file
 -d,--default         Generate a configuration file
 -E,--end <arg>       End revision, for incremental (SVN)
 -e,--extract <arg>   Extract a revision
 -i,--info            Report on repository usage
 -r,--repo <arg>      Repository file/path
 -S,--start <arg>     Start revision, for incremental (SVN)
 -t,--type <arg>      SCM type (CVS | SVN)
    --tags <arg>      find tags to specified depth
    --tree <arg>      (with --info), display tree to specified depth
 -u,--users           List repository users
 -v,--version         Version string

Example: standard usage.
   java -jar p4convert.jar --config=myFile.cfg

Example: generate a CVS configuration file.
   java -jar p4convert.jar --type=CVS --default

Example: report Subversion repository usage.
   java -jar p4convert.jar --type=SVN --repo=/path/to/repo.dump --info
```

To create a default configuration file using the ***--default*** option. You will need to specify the SCM type using ***--type*** (with a value of SVN or CVS). The generated file (default.cfg) will contain a set of default configuration options based on your platform. Rename the ***default.cfg*** to your own configuration. To run a conversion using your configuration file, execute the jar with the ***--config*** flag specifying you config file. For example:

```
java -jar p4convert.jar --config=myFile.cfg
```

Generating a Subversion Dump file

For Subversion conversions, both Import and Convert modes require a Subversion dumpfile as the historic data source. A dumpfile can be generated in several different ways, two of which are detailed below:

- Using the svnadmin command:

```
svnadmin dump local_repo_path > dumpfile.dmp
```

  Replace ***local_repo_path*** with the path to the Subversion repo. Do not use the ***--delta*** flag option as the results cannot be parsed by the conversion tool.

- Using the remote dump command:

```
rsvndump url > dumpfile.dmp
```

  Replacing ***url*** with the remote Subversion server. For example: **http://** or **file://** or **svn://**. Note that for large servers generating the dump file locally is significantly faster, so you might also consider creating a local copy of the subversion depot using svnsync, before generating the dump file.

Selecting a CVS root

For CVS conversions, both Import and Convert modes require a path containining RCS files as the historic data source. CVSROOT can be filtered or imported on a per module basis. The converter will only import file revisions under the specified CVSROOT path.

Chapter 2

Configuration

The java conversion requires a configuration file to setup the necessary options. A default configuration file `default.cfg` can be generated by running the converter with the *`--default`* parameter and specifying *`CVS`* or *`SVN`*.

For example: to generate a default configuration file for CVS, run...

```
java -jar p4convert.jar --type=CVS --default
```

Once you have `default.cfg`, copy it to `yourconfig.cfg` and then customize it as needed for the conversion.

Core converter settings.

The following settings appear at the top of the configuration file and are automatically defined at the time of generation. The Core converter settings should not be modified.

1. Configuration file schema version. Used to identify older configuration formats.

```
com.p4convert.core.schema=5.50
```

2. SCM conversion type. Either set SVN or CVS configuration options.

```
com.p4convert.core.scmType=SVN
```

3. Reserved for testing framework. Must be set to `false` for normal operation.

```
com.p4convert.core.test=false
```

4. Conversion tool's release version.

```
com.p4convert.core.version=PUBLIC.10690
```

General Perforce converter settings.

1. Set the conversion mode to `IMPORT` for Import Mode and to `CONVERT` for Conversion Mode.

```
com.p4convert.p4.mode=IMPORT
```

2. Import depot path and depot to be created, e.g. `//import/...`

```
com.p4convert.core.depotPath=import
```

3. An optional sub path can be specified below the depot, e.g. `//import/sub/...` (Note that the path must end with a slash (`/`) even if no path is used). The default value is `/`.

```
com.p4convert.core.subPath=sub/
```

General Subversion converter settings.

If the configuration file has been correctly generated using the *--type=SVN --default* the Subversion configuration options will appear as a block using the *com.p4convert.svn* name space. These settings will only apply if the SCM mode is set to *SVN*:

```
com.p4convert.core.scmType=SVN
```

1. Local path to Subversion dump file.

```
com.p4convert.svn.dumpFile=/Users/bruno/MyDumpFile.dmp
```

General CVS converter settings.

If the configuration file has been correctly generated using the *--type=CVS --default* the CVS configuration options will appear as a block using the *com.p4convert.cvs* name space. These settings will only apply if the SCM mode is set to *CVS*:

```
com.p4convert.core.scmType=CVS
```

1. Local path to CVSROOT files. (A deeper path is permitted to limit the scope of the import).

```
com.p4convert.cvs.cvsroot=/Users/bruno/CVSROOT/
```

2. (Optionaly) specify a CVS module to import from within the CVSROOT.

```
com.p4convert.cvs.cvsmodule=projX
```

3. CVS time window. When no *commitid* is defined in the RCS file, the converter will calculate which revisions should be grouped into a changelist. The time window specifies the maximum range the converter should look ahead for matching revisions. (milliseconds)

```
com.p4convert.cvs.timeWindow=20000
```

4. CVS temporary directory. All the CVS assets are reformatted into full text files. The specified temporary directory is used to store the content during conversion.

```
com.p4convert.cvs.tmpDir=tmp
```

Subversion: Selective and Incremental Conversions

1. The Subversion revision ranges can be limited using the configuration options ***startRevision*** and ***endRevision***. The default is to start at revision 1 and import all revisions, defined by setting ***endRevision*** to 0.

```
com.p4convert.svn.start=1
com.p4convert.svn.end=0
```

2. Incremental conversions are possible using the Import Mode. It is important to take note of the last imported Subversion revision and start from the next revision for the subsequent import. For example: to import the first 100 subversion revisions, set the config options to:

```
com.p4convert.svn.start=1
com.p4convert.svn.end=100
```

3. At the start of the import the following summary is displayed: (In this example only the first 100 revisions are imported out of 23000.)

```
Importing Subversion ranges:
  start: 1
  end:   100
  last:  23000
```

4. For the next incremental import (say another 100 revisions) set the revision ranges to...

```
com.p4convert.svn.start=101
com.p4convert.svn.end=200
```

5. The start and end revisions can be override on the command line using the ***--start*** and ***--end*** flags. For example, importing just Subversion revision 201.

```
java -jar convert.jar --config=default.cfg --start=201 --end=201
```

6. In adition to overriding the start and end revisions you can override Subversion dump file location using the ***--repo*** flags. For example, a range of revisions from an incremental dump.

```
svnadmin dump /path/to/repo/ -r 202:300 --incremental > 202-300.dump

java -jar convert.jar --config=default.cfg --repo=202-300.dump --start=202 --end=300
```

Filtering Subversion paths

The default behavior is not to exclude any path in Subversion; however for large repositories with many `tags/` folders or situations where only part of a Subversion repository is to be

converted you may wish to exclude certain Subversion paths. Subversion path exclusion is possible using two map files `exclude.map` and `include.map`.

The filtering is based on matching the Subversion path to a regular expression in the map files. The `exclude.map` file is processed first and if the pattern matches part of the path then that path is skipped. The `include.map` file can be used to overlay the `exclude.map` file re-adding paths that were skipped. Before a conversion can start the filters must be verified against the Dump file (typically this is fairly quick and a displays a progress indicator).

The verification step is to prevent the situation where an excluded path is relied on at a later point in the history for a branch, copy or merge action. If such a situation is found the paths are logged and the excluded *`source path`* is added to the `issue.map` file. To resolve the issue the exclusion should be removed from the `exclude.map` file or a regular expression, based on the `issue.map` file, added to the `include.map` file.

For example; to exclude all Subversion tags in the folder 'tags/', create an exclusion map file `exclude.map`:

```
# exclude Subversion tags:
^tags/.*
```

Then start the conversion to verify the map:

```
pallen-mac:main$ java -jar dist/p4convert.jar --config=Config/foo.cfg
loading ChangeMap:  changeMap.txt
loading TypeMap:  types.map
importing revisions:  1 to 20635 out of 20635
exclude.map: ^tags/.*
Verifying exclusion map...
issue: tags/rel-1.0.14/api
issue: tags/rel-1.0.14/sys
issue: tags/rel-2.0.3
Issues found, saving issue map...
Caught EXIT shutting down ...
```

Looking at the reported issues the tags 'rel-1.0.14' and 'rel-2.0.3' have some actions that conflict with our exclusion, to resolve this simply add the exclusions to the 'include.map' file:

```
# issues reported for tags/ folder
^tags/rel-1.0.14/.*
^tags/rel-2.0.3/.*
```

Labeling Subversion Paths

Whilst Subversion does not have a specific label entity may users follow the 'tags' convention. If a 'tag' path is pure (in that it has not had subsequent changes to the content) it can be converted to a Perforce label.

1. To enable labeling of Subversion tags, set the *`com.p4convert.svn.labels`* to *`true`* and define maps: `exclude.map` and `include.map`.

```
com.p4convert.svn.labels=true
```

2. The maps are designed with resect to branches. The default behaviour is for all Subversion tag operations to be imported as Perforce branches. In order to change the behaviour and attempt to import the Subversion tag as a Perforce label the path must be in the exclude map.

   For example; to import all Subversion tags in the folder 'tags/' as Perforce labels, create an exclusion map file **exclude.map**:

```
# exclude Subversion tags:
^tags/.*
```

3. A Perforce label will need a unique name space derived from the 'tags' path. The conversion tool provides a depth and regular expression configuration option to assist.

```
com.p4convert.svn.labelFormat=svn_label:{depth}
```

```
com.p4convert.svn.labelDepth=2
```

   The depth determins how much of the path to use for the unique label name. Typically a value of '2' for convertional Subversion usage. With a depth of '2' the first two elements of the 'tags' path are then held in an array and used by a regular expression to generate the label name.

   For example, given a Subversion tag located **tags/1.0.0/** the following regular expressions provide the coresponding label names:

```
com.p4convert.svn.labelFormat=label:{2}         label:1.0.0
com.p4convert.svn.labelFormat={1}-{2}           tags-1.0.0
```

Note An empty expression will use the original path.

```
com.p4convert.svn.labelFormat=                  tags/1.0.0/
```

Note If a keyword of ***{depth}*** is used it will be substituted with it's value.

```
com.p4convert.svn.labelFormat=label_{depth}     label_1.0.0
```

Changelist Offset options

1. The default offset is `0` (i.e. not offsetting). Offset is currently available for Convert Mode and allows subversion revisions to be converted into Perforce changelists by a fixed offset:

```
com.p4convert.p4.offset=0
```

2. After the conversion a *changeMap* file is appended locally containing a Subversion revision number to Perforce change-list mapping. The file name is configured by the option:

```
com.p4convert.log.changeMap=changeMap.txt
```

   A changemap looks like this:

```
# <Change>, <SVN revision>
1, 1
2, 2
3, 3
...
```

Unicode Support

The following Unicode enable options apply to Unicode support for Import Mode and Convert Mode. The Charset options are only applicable to Import Mode, when translating a file through the Perforce client. In Convert Mode archive files are always written in UTF-8 for a Unicode enabled Perforce server.

Defaults (for non-Unicode servers):

```
com.p4convert.p4.unicode=false
com.p4convert.p4.charset=<none>
```

A UTF8 client example: (for a full list see `p4 help charset`)

```
com.p4convert.p4.unicode=true
com.p4convert.p4.charset=utf8
```

For Unicode conversions set the JVM arg:

```
-Dfile.encoding=UTF-8
```

Some Solaris and Linux conversions may need the locale set:

```
export LC_ALL=en_GB.UTF-8
```

Once a Perforce server is switched to Unicode enabled mode (`-xi`), all client workspaces need to define a character set. For details see:

http://answers.perforce.com/articles/KB_Article/Internationalization-and-Localization

Note	A non-Unicode enabled Perforce Server will accept UTF16 files.

Normalisation

Platform Unicode normalisation is detected when the configuration file is generated, however it can be changed by setting the following configuration option to `NFC` or `NFD`:

```
com.p4convert.p4.normalisation=NFD
```

The default detection is based on the following:

Platform	Normalization
Windows	`NFC`
Mac	`NFD`
*nix/*nux	`NFC`
Sun	`NFC`

Subversion Properties

By default, the converter parses Subversion properties as UTF-8 strings. The conversion uses properties such as `svn:log`, `svn:author` for attributes in Perforce and must decode the byte sequence to UTF-8. In some data sets Windows users may have added high ASCII characters in one or more code pages. This release now supports a configuration option:

```
com.p4convert.svn.propTextType=UNKNOWN
```

The following strings denote the supported char-sets:

Big5	IBM424_rtl	ISO-8859-7	UTF-16LE
BINARY	ISO-2022-CN	ISO-8859-8	UTF-32BE
EUC-JP	ISO-2022-JP	ISO-8859-9	UTF-32LE
EUC-KR	ISO-2022-KR	KOI8-R	UTF-8
GB18030	ISO-8859-1	Shift_JIS	windows-1251
IBM420_ltr	ISO-8859-2	UNKNOWN	windows-1252
IBM420_rtl	ISO-8859-5	US-ASCII	windows-1254
IBM424_ltr	ISO-8859-6	UTF-16BE	windows-1256

The first scan is always `UTF-8` followed by the configuration option. `BINARY` implies a skip and the string `<binary property>` is inserted.

Advanced Configuration

Directory Properties

The following options allow Subversion Directory Properties to be stored as versioned files in Perforce. To enable this mode set the following property to `true`:

```
com.p4convert.svn.propEnabled=true
```

To select the property name and format: (Note: only `ini` mode is supported)

```
com.p4convert.svn.propEncoding=ini
com.p4convert.svn.propName=.svn.properties
```

Empty changelists

The following property will attempt to skip empty changes (where the change contains no revisions). This is typically the default behavior of the client or Import Mode, so it is only really used in Convert Mode.

```
com.p4convert.p4.skipEmpty=false
```

Username translation

A username map file (`users.map`) can be generated using the *`--users`* option and then the right-hand-side modified with the new user name. The rename will only occur if the conversion tool finds the `users.map` file in the current working directory. Username mapping is not currently supported for CVS conversions.

Binary file detection

Binary files can be identified by adding their extensions to the type map file `types.map`. The format is based on Perforce typemap spec, however it is limited to paths of the form `//....`*`ext`* (where *`.ext`* is the binary extension).

Default Type map (`types.map`):

```
TypeMap:
        binary //....zip
        binary //....gif
        binary //....png
        binary //....jpg
        binary  //....dll
        binary  //....class
        binary  //....jar
        binary  //....ecsfr
```

Modification bits (`+mxwlk`) are supported and can be added using the type mapping. Binary detection though the type map is recommended as conversion is much faster. Binary files not identified in the type map will be scanned by the ICU4J libraries and if no text/Unicode match is found they are assumed to be binary.

Warning ICU4J may incorrectly identify small binary files as text creating sync issues on Windows clients.

Changelist Description Format

The *`logRevID`* option can be used to reformat the Subversion revision descriptions to include the revision ID using the template:

- *`><rev>`* substituted with the Subversion revision
- *`<description>`* substituted with the Subversion log

The default value (as-is):

```
com.p4convert.p4.logRevID=<description>
```

Case Sensitivity

The platform case sensitivity is detected when generating the configuration file. There is normally no reason to change this behavior from the detected defaults. Conversions between different platforms should be avoided especially when converting from a case sensitive environment (such as Linux) to a case insensitive environment (such as Windows). The advanced case handling options supported are set using one of the following options:

```
com.p4convert.p4.caseMode=FIRST
```

- `NONE` - treat all paths as case sensitive (Linux).
- `LOWER` - convert all paths to lower case
- `UPPER` - convert all paths to upper case
- `FIRST` - use the first encountered case combination (Windows)

When using Convert Mode the generated Perforce archive files are based on the platform's case sensitivity. However on Linux platforms it can be useful to store archive files as if on a case-insensitive server (`p4d -C1`). This can be simulated by setting the following option to `true`:

```
com.p4convert.p4.lowerCase=true
```

Important If this option is set the path to the Perforce root directory, defined by *`com.p4convert.adv.p4root`*, must be in **lower case** and the case mode of `FIRST` must be used.

RCS Keyword expansion (svn:keywords)

By default, RCS keyword expansion attributes are imported; however setting the configuration option:

```
com.p4convert.svn.keepKeyword=true
```

Setting a value of `false` will ignore all previous keyword attributes and import the files as normal text. See keyword notes for known issues.

Merge Information (svn:mergeinfo)

Supports Subversion 1.5-1.7 merge information and calculates the corresponding Perforce integration credit for the various actions. The feature is not enabled by default and if required the following configuration option must be set to true:

```
com.p4convert.svn.mergeInfoEnabled=true
```

Running P4Convert

Import Mode

Import Mode will pull in file revisions from a Subversion dump file or CVS repository, adding them to the Perforce Server specified by the connection details in the configuration options. To use this mode set:

```
com.p4convert.p4.mode=IMPORT
```

Care should be taken when adding data to a pre-existing Perforce Server that the revision actions do not conflict with revisions already in the Perforce Server; typically, this can occur if the Server has been in-use since the previous migration.

To avoid such scenarios the import should either be to a unique depot, to avoid conflict, or the Perforce Server should be Read Only during subsequent migrations. After a conversion is complete, it is possible to merge the new data with an existing depot using tools like PerfMerge++.

The converter will check that your Perforce Server has no pending changes, and will abort a conversion if any are detected.

Configuration options specific to Import Mode

1. Perforce server address and port (escape the '**:**' with '**\:**'):

```
com.p4convert.p4.port=localhost\:4444
```

2. Default user and client for server connection:

```
com.p4convert.p4.client=p4-client
com.p4convert.p4.user=p4-user
```

3. Client workspace root used to import files into Perforce:

```
com.p4convert.p4.clientRoot=/Users/bruno/ws/
```

4. If Security is set to level 1 or greater, then the ***p4-user*** must have *admin* permissions in the Protection table and the password supplied as a string (the default is set to **ws/** under your local directory):

```
com.p4convert.p4.passwd=PaSSwoRd
```

5. Alternatively if the user is already logged in and there is a valid ticket, then leave the password field unset and set your environment for **P4TICKETS** (don't rely on the **P4TICKETS** unset default as **p4-java** will not find your ticket file).

Convert Mode

Convert Mode is more advanced and requires knowledge of the Perforce Journal replay and Archive file store. This mode can only be used for single shot conversions and cannot be used incrementally. The performance of Convert Mode is significantly better than Import Mode (x100 sometimes!).

After using Convert Mode the administrator will need to run several commands to rebuild the Perforce server and upgrade the metadata. Please refer to the "Post Conversion [Conversion Mode]" on page 17 section for step-by-step instructions.

To use this mode set:

```
com.p4convert.p4.mode=CONVERT
```

Configuration options specific to Conversion Mode

1. Perforce server root address (path should end with a slash (/)), the default is set to `p4_root/` under you local directory:

   ```
   com.p4convert.p4.root=/full/path/to/p4_root/
   ```

2. Change list offset from which to start conversion (handy for batched conversions):

   ```
   com.p4convert.p4.offset=0
   ```

3. Generated journal names (useful to increment the prefix when running batched conversions):

   ```
   com.p4convert.p4.jnlIndex=0
   com.p4convert.p4.jnlPrefix=jnl.
   ```

4. Mimic the 2011.1 or greater credit behavior on rollbacks / downgrades (to enable set value to `true`):

   ```
   com.p4convert.p4.downgrade=false
   ```

5. Perforce normalises line-endings when storing the file on the server and restores them based on the client workspace options and platform type. However, in special cases it can be useful to store line-ending in the server and use the 'share' option in the client. To disable normal line-ending support set the following option to `false`:

   ```
   com.p4convert.p4.lineEnding=true
   ```

6. For non Unicode servers, or to simplify storage of hi-ASCII files, setting the following option to `false` will store the file as `binary`:

```
com.p4convert.p4.unicode=false
```

Post Conversion [Conversion Mode]

To finish a Convert Mode conversion you will need to install `p4` and `p4d` and run a few Perforce commands.

1. **[Required]** Change directory to `P4ROOT`, check there are no `db.*` files present and then replay the journal file(s):

```
$ cd p4_root
$ p4d -r . -jr jnl.0
Perforce db files in '.' will be created if missing...
Recovering from jnl.0...
Perforce server info:
 Server version 33 is replaying a version 0 journal/checkpoint.
```

Note

The Server version is set to `0` to remind the administrator that an upgrade is required, see step 3.

2. or, for multiple journal files:

```
$ p4d -r . -jr jnl.0 jnl.1 ....
```

and with nohup:

```
nohup p4d -r . -jr jnl.0 jnl.1 .... &
```

3. **[Required]** Upgrade the database from 2004.2 schema. For simplicity the conversion generates a database using an old schema, allowing you to upgrade to a Perforce Server version of your choice. From the `P4ROOT` directory run the upgrade command:

```
$ p4d -r . -xu
Perforce db files in '.' will be created if missing...
2001.1: splitting db.integ into db.integed/db.resolve.
2001.1: splitting db.have into db.have and db.label.
2002.1: splitting pending db.change into db.changex.
2002.2: upgrading tempobj filetype in db.rev.
2002.2: upgrading tempobj filetype in db.working.
2003.1: initialize default depot.
2003.2: upgrading db.user.
2005.1: building db.revhx (headrev) table.
2005.1: building db.locks from db.working.
2005.2: building db.revdx (delrev) table.
2005.2: moving spec depot entries into db.revsx.
2007.3: (re)building haveMap from db.have/db.working.
2007.3: (re)building db.archmap (lazy-copy map) table.
2007.3: removing old db.archive.
2008.1: upgrading db.change.
2009.2: moving db.boddate/db.ixdate into db.bodtext/db.ixtext.
2009.2: removing db.boddate/db.ixdate.
2010.2: adding db.config.
2011.1: upgrading tiny.db.
...upgrades done
```

4. [Optional] If the conversion was run in Unicode mode (where `com.p4convert.p4.unicode=true`) and users are going to continue to add Unicode content then you may wish to set the server to Unicode mode. To enable Unicode run the following command from the P4ROOT directory:

   ```
   $ p4d -r . -xi
   ```

5. [Alternative] If you are running your Perforce Server on Linux and your user base is predominantly Windows you may wish to force the server to run as case insensitive (only allowing one version of case for paths and files). The conversion option `com.p4convert.adv.lowerCase=true` will have been used with the case mode set to `com.p4convert.adv.caseMode=FIRST`. All p4d commands must be include the `-C1` flag, this includes the earlier step 1 and step 2.

   For example:

   ```
   $ cd p4_root
   $ p4d -C1 -r . -jr jnl.0
   $ p4d -C1 -r . -xu
   ```

6. [Recommended] Some archive files may not have MD5 sum digests (typically where Subversion did not store the digest or the digest does not match due to the use of keyword expansion). To fill in the metadata for MD5 sum digests and archive file sizes use the verify command (this might be best split into depots and sub directories for large servers):

   ```
   $ p4 verify -u //...
   ```

Chapter 4

Notes

Keyword Expansion Issues:

The conversion process preserves keyword file types and sets these files in Perforce with the +k modifier. However keywords in Subversion and Perforce are expanded differently. For example: In Subversion the keyword **$Revision$** expands to:

```
$Revision: 25005 $  (a change number in Perforce)
```

Another example is the keyword **$Date$**, which in Subversion gives you the time as well...

```
$Date: 2006-02-17 12:09:10 +0000 (Fri, 17 Feb 2006) $
```

and in Perforce...

```
$Date: 2006/02/17 $
```

Here is a list of alternatives, (based on the above example):

```
$Revision$    ==> $Change$ gives $Change: 25005 $
$Date$        ==> $DateTime$ gives $DateTime: 2006/02/17 12:09:10 $
```

Due to the differences in keyword expansion the MD5 sum in Subversion is not valid for use in Perforce. This is why any keyword expanded files have an empty MD5 sum and why the **p4 verify -u //...** command is recommended in the 'Optional Steps' section.

Verification

Verification can be performed using a running Subversion server and comparing the differences in files in Perforce using the audit log. To enable the audit log set the following option to **true** and choose the file name.

```
com.p4convert.log.audit.enabled=true
com.p4convert.log.audit.filename=audit.log
```

A sample line of the audit log:

```
# <SVN path>, <SVN revision>, <P4 change>, <MD5 sum>
trunk/src/foo.c, 1, 1, 1234567890abcdef1234567890abcdef
```

Output and Logs

Console output and logging configuration options

The default SLF4J logging options can be overwritten with another configuration file; specify the ***log4j.configuration*** option with a local file, using the syntax:

`file:`*`your_local_file`*

```
java -Dlog4j.configuration=file:log4j.xml -jar p4convert.jar
```

Please use or refer to the sample **`debug.log4j.xml`** for logging options.

These options are reserved for testing or future enhancements:

```
com.p4convert.core.test
com.p4convert.core.version
com.p4convert.svn.emptyDirEnabled
com.p4convert.svn.emptyDirName
```

Reading Console/Logging output

```
Importing Subversion ranges:

start: 1
end:   23091
last:  23091

1.0 A:D - trunk
1.1 A:D - trunk/src
1.2 A:F - trunk/src/foo.c (UTF-8)
mapping: r1 => @1

2.0 A:F - trunk/src/bar.png (BINARY)
mapping: r2 => @2
...
```

Explanation of output lines, for example:

```
1.2 A:F - trunk/src/foo.c (UTF-8)
```

The numbering **`1.2`** refers to the current Subversion revision and the node action. (**`1`** Subversion revision 1 and **`.2`** = the third node action as **`.0`** would be the first index)

The letters **`A:F`** refers to the Subversion action and if it is a file or directory operation.

- A Add
- B Branch
- E Edit
- I Integrate (merge fromNode)
- M Merge (svn:mergeinfo)
- C Copy (svn replace action with fromNode)

- U Update (svn replace action)
- R Remove (delete)
- F File
- D Directory

The subversion path `trunk/src/foo.c` is followed by the detected type.

Detected Type	Perforce Base Type
UTF-8	Unicode
UTF-16BE	utf16
UTF-16LE	utf16
UTF-32BE	Unicode
UTF-32LE	Unicode
Shift_JIS	Unicode
ISO-2022-JP	downgraded to binary
ISO-2022-CN	downgraded to binary
ISO-2022-KR	downgraded to binary
GB18030	Unicode
EUC-JP	Unicode
EUC-KR	Unicode
Big5	Unicode
ISO-8859-1	Unicode
ISO-8859-2	Unicode
ISO-8859-5	Unicode
ISO-8859-6	downgraded to binary
ISO-8859-7	Unicode
ISO-8859-8	Unicode
windows-1251	Unicode
windows-1254	Unicode
windows-1256	downgraded to binary
KOI8-R	Unicode

Detected Type	Perforce Base Type
ISO-8859-9	Unicode
IBM424_rtl	downgraded to binary
IBM424_ltr	downgraded to binary
IBM420_rtl	downgraded to binary
IBM420_ltr	downgraded to binary
BINARY	binary

Note The detected type is based on language detection using ICU4J and not Subversion MIME or Perforce detection.

Finally `mapping: r223 => @223`' refers to the Subversion revision number (`r223`) to the Perforce changelist number (`@223`). Typically 1:1 unless offset or merged against an live or pre-existing Perforce Server.

Errors

In Import Mode, Perforce related `p4-java` messages are reported as warnings and should be verified, for example:

```
46.3 A:F - repo/trunk/my.file
WARNING: p4java: //import/repo/trunk/my.file - file(s) up-to-date.
```

In Conversion Mode, errors are reported as *Panics* and an exception is thrown.

Problematic Subversion dump records can be extracted and sent to Perforce when it is not possible to send in the whole dumpfile. The extracted dumpfile only contains meta-data and the file content is removed and replaced with a block count.

During an exception note the Subversion revision and node ID (the example above has a revision number of 46 and a node number of 3). Then run the following command to extract the problematic record, for example:

```
$ java -jar dist/p4convert.jar --type=SVN --repo=mysvndump.dump --extract=46.3

searching for node: 46.3...
Node-path: repo/trunk/my.file
Node-kind: file
Node-action: add
Prop-content-length: 10
Text-content-length: 308
Text-content-md5: 6a339b6ccf2af72d77169ef29b98eb0b
Content-length: 318


PROPS-END
```

A file is then generated called `node.46.3.dump` and will contain the Subversion record to be sent to Perforces, as well as the Subversion meta-data (everything after `searching for node: 46.3...`).