INTERSYSTEMS LEARNING SERVICES

InterSystems Change Control

InterSystems
Creative data technology

InterSystems Change Control

COPYRIGHT NOTICE

© 2020 InterSystems Corporation, Cambridge, MA. All rights reserved.
InterSystems is a registered trademark of InterSystems Corporation.

InterSystems, InterSystems Caché, InterSystems Ensemble, InterSystems HealthShare, HealthShare, InterSystems TrakCare, TrakCare, InterSystems DeepSee, and DeepSee are registered trademarks of InterSystems Corporation.

InterSystems IRIS data platform, InterSystems IRIS for Health, InterSystems IRIS, InterSystems iKnow, Zen, and Caché Server Pages are trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Congress Street, Boston, MA 02114, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems WorldWide Response Center
Telephone: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

ICC 460: CCR Transport – Perforce Debugging & Advanced Topics

InterSystems
Creative data technology

Objectives

- Be able to use p4V and Swarm to make changes directly to items on the Perforce server.
- Define view paths and how they are used within Perforce.
- Describe Perforce command line interface variables, how they are used and setup.
- Explain how Perforce references files during synchronisation.
- Illustrate how Perforce prevents conflicts using locks.
- Leverage branch hygiene tool.

Objectives (cont.)

- Setup Perforce notifications to monitor specific branches.
- Demonstrate how to move a changelist to a new CCR using Swarm and P4V.
- Explain how to refresh Perforce branches.
- Demonstrate how to backout changes using Perforce.
- Fix integration history.

Part 1: Advanced Perforce Concepts

View Paths

- View path: //depotMapping/... //targetMapping/...
  - Examples:
    - A Client Workspace View Path between a depot and a client: //MYDEPOT/MyApp/MAIN/... //MYDEPOT_MyApp_MAIN_Dev/...
    - A Branch View Path between two depot codelines: //MYDEPOT/MyApp/MAIN/....//MYDEPOT/MyApp/BRANCHES/1.0.0/...
- Overlay & Exclusions – used to remove or add specific locations to a View Path.
  - +//mapping3/... //mapping4/...
  - -//mapping3/... //mapping5/...

View Paths (cont.)

- Special characters for View Paths: space, *, ..., %, @, #.
  - Perforce wildcards – avoid in filenames.
  - Use ASCII code or quote.

CLI: P4 Session Variables

- To use the CLI (Command Line Interface) it is a good idea to set up your CLI variables:
  - P4CONFIG = p4config.txt (Windows) ; .p4 (UNIX).
    - P4PORT=server:port.
    - P4CLIENT=clientName.
    - P4USER=lanID.
    - P4CHARSET= utf8 (Optional).
- If not logged, must issue “p4 login” command first.
- Always check where you are connected with “p4 info” before issuing commands by the CLI.

CLI: Perforce Session

- Hierarchy of elements defining your P4 environment and command settings:
  - On the command line, using flags.
  - In a configuration file, if P4CONFIG is set.
  - User environment variables (on UNIX or Windows).
  - System environment variables (on Windows, system-wide environment variables are not necessarily the same thing as user environment variables).

CLI: Perforce Session (cont.)

- Hierarchy of elements defining your P4 environment and command settings:
  - On Windows, in the Perforce User Registry:
    - p4 set setting=value.
  - On Windows, in the Perforce System Registry:
    - p4 set -s setting=value.
- You can check your environment variables with “p4 set” before issuing commands with the CLI.

Working on Files: Synchronizing

- p4 sync.
- @, # are the wildcards to use to specify versions:
  - @ = relative reference.
    - changelist.
    - date.
    - Label.
    - etc…
  - # = hard reference (specific revision).
    - Useful for fetching specific revision of one file.
    - Not very useful for group of files.
    - #head specifies the ‘latest’ revision.

Preventing Conflicts: Lock

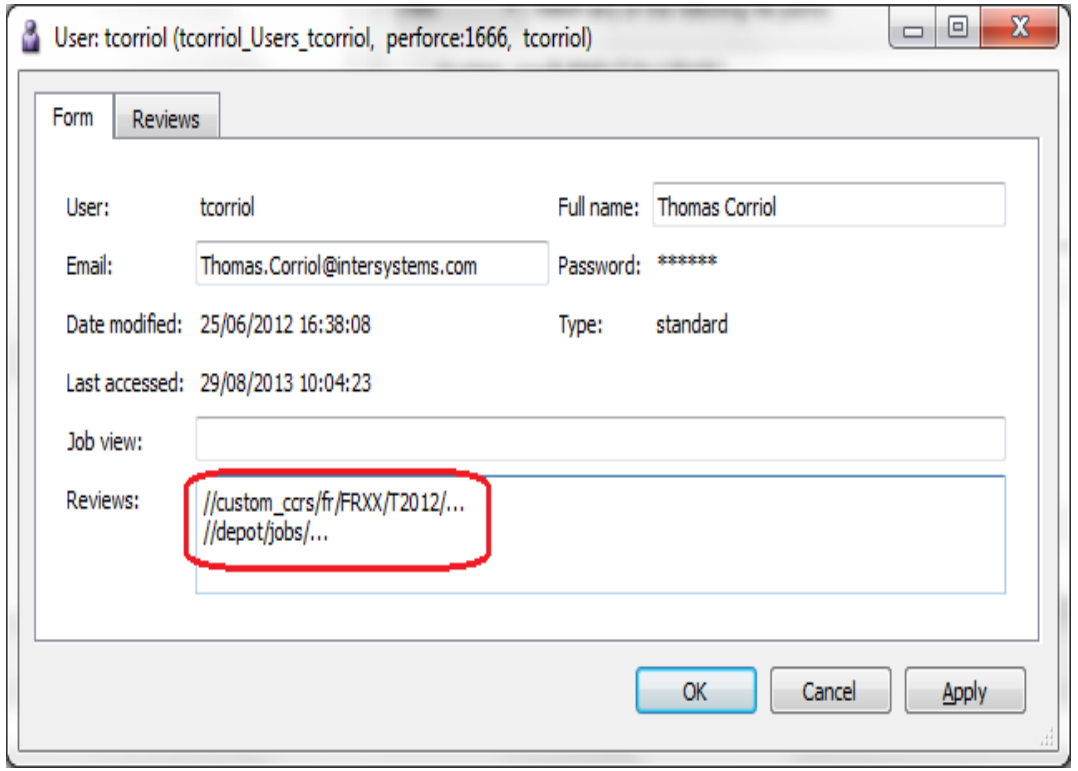
- Optimistic locking.
  - Use Lock command in P4V.
  - Prevents other user checking in the file before you check in the file.
- Pessimistic locking.
  - Handled on the server via specific filetypes.
  - Prevents other user from checking out the file before you check in the file.
    - It is helpful to use pessimistic for certain files types, e.g. PPT and XLS files which can’t be merged.
  - Contact Tech Services if there is a specific file type for which you want pessimistic locking in your depot.

Performing Very Large Branching Virtually

- There are some cases where a large number of files need to be branched to a new location.
  - E.g., populating environment branches after a CCR Baseline, or making a new Project Branch for a product.
- By default, all files get synced to the local workspace as part of an integration, which can be very time consuming.
  - Especially on remote connections.
- To branch large volumes of files from depot to depot without syncing them locally:
  - Use the -v switch for p4 integrate, which will perform the branching virtually and will be much faster.

Perforce Notifications via “Reviews”

- Changelists notifications via email can be set up to ‘watch’ a specific branch.
  - Allows specific branches to be ‘monitored.’
  - Configurable by user.
  - Menu “Connection > Edit Current User...”
    - Put a branch under “Reviews.”



Perforce Notifications via “Reviews” (cont.)

- Changelists notifications via email can be set up to ‘watch’ a specific branch.
  - When a changelist is submitted against a branch, an email gets sent immediately with:
    - User.
    - Changelist description.
    - Files impacted.
    - Job.
  - “//depot/jobs” to get jobs notification (you will get ALL of them).
  - Metadata is not protected, and shared on a Server.

Quiz: Advanced Perforce Concepts

Question:

If the Perforce system default for check-outs is optimistic locking, both John and Sandy can checkout the same file.

True or False?

Answer: True.

If Sandy wants to prevent John or anyone else from checking out the file, she should use the lock command after checking out the file.

Part 2:
Moving a Cancelled Changelist to a new CCR

Moving a Changelist from One CCR to Another

- There are some cases where a changelist needs to be moved from one CCR to another:
  - An ItemSet was uploaded against the wrong CCR by accident.
  - A CCR was cancelled by mistake (and the changelists were not backed out) and the user wishes to attach the changelist to a new CCR to progress the change.
- Review: changelists are associated with CCRs via Perforce jobs.

Moving a Changelist from One CCR to Another (cont.)

- Jobs can be edited on submitted changelists via:
  - Swarm.
  - p4V.
- Adding a new job (which references a CCR) can be done only by InterSystems employees as it requires direct access to Perforce.
  - Customers should contact InterSystems Support for assistance in moving a changelist.

Using Swarm to Change a Job on a changelist

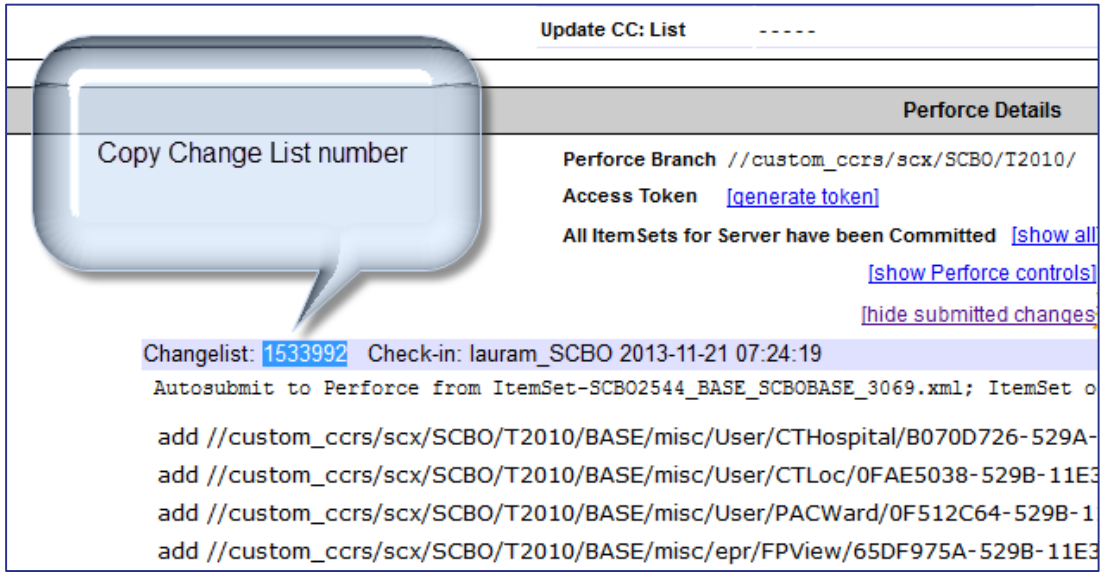
- Click the changelist number on the CCR that you need to move, then it will open it up in Swarm (using your Perforce credentials).
- Add the correct CCR number by clicking the "Add Job" hyperlink.
- Remove the link to the incorrect one by clicking the X beside it.

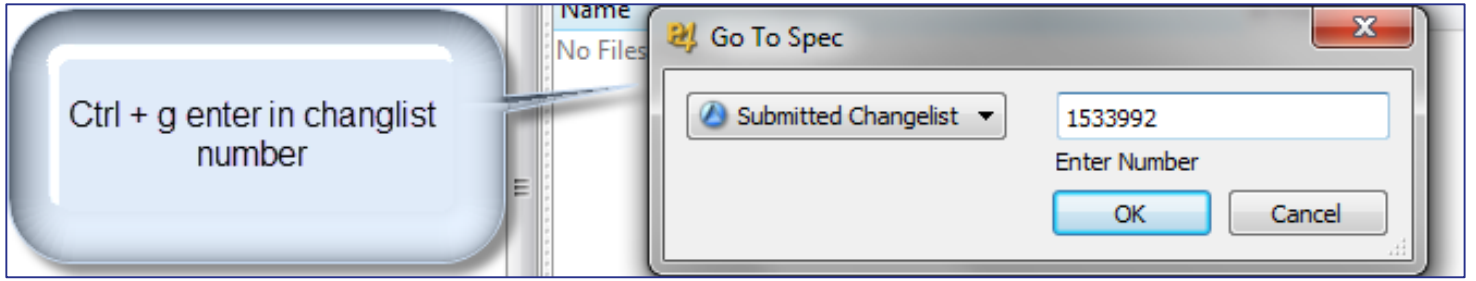
Using p4V to Change a Job on a Changelist

- Steps:
  1. Log into CCR online application.
  2. Ensure that the new CCR is in a phase of In_BASE.
  3. Open the CCR with which the changelist is currently associated.
  4. Select Show Submitted Changes hyperlink within the Perforce Details Section.

Using p4V to Change a Job on a Changelist (cont.)

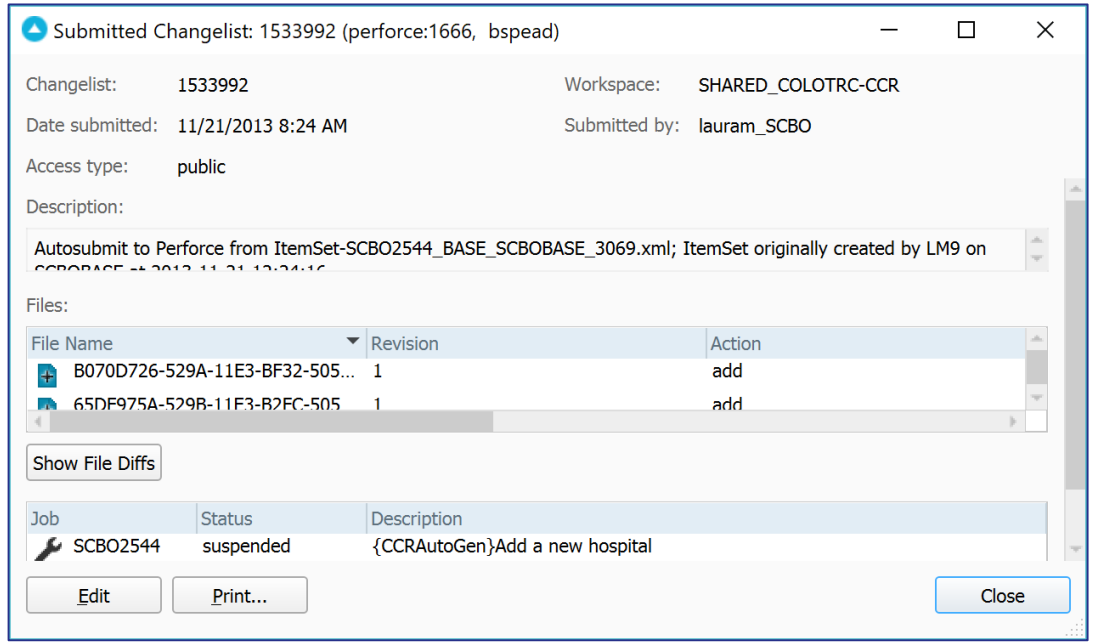
- Steps:
  5. Copy the changelist Number.
  6. Log into p4V.
  7. Open the changelist (CTRL+G).
  8. Paste the changelist number.
  9. Click on OK.

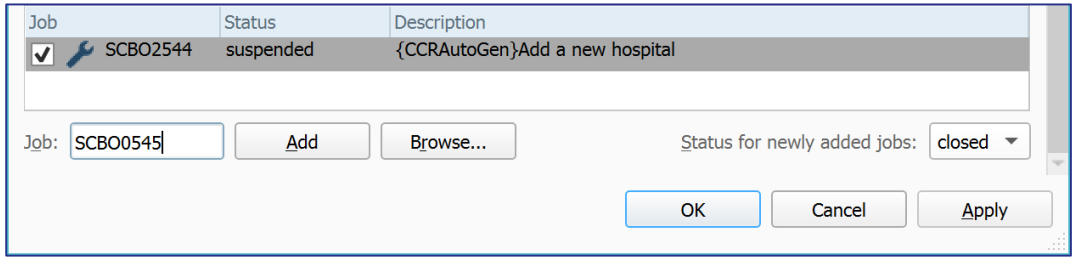




Using p4V to Change a Job on a Changelist (cont.)

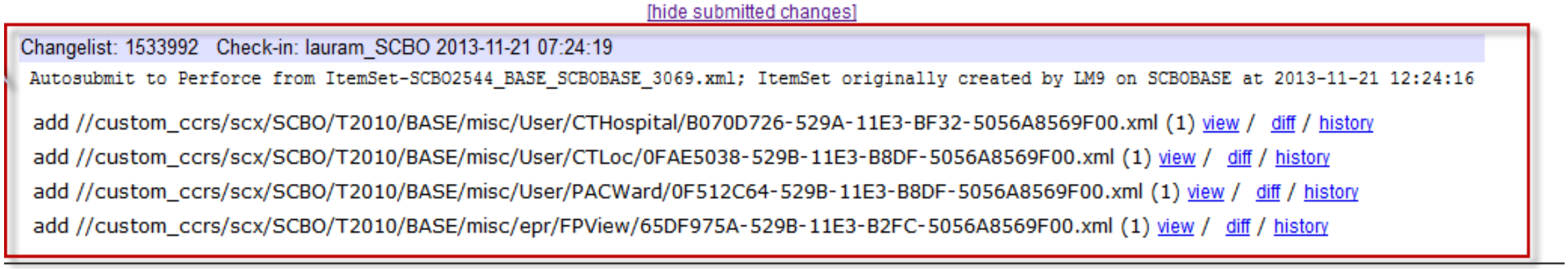
- Steps:
  10. Click Edit.
  11. Enter new CCR number in Job field.
  12. Click Add.
  13. Confirm the job is now listed.
  14. Right-click the old job > Remove from changelist.
  15. Click OK.





Using p4V to Change a Job on a Changelist (cont.)

- Steps:
  16. Return to new CCR and refresh page.
  17. Verify changelist now appears under [show Submitted Changes].
  18. Progress CCR according to normal workflow.



Quiz: Moving Cancelled Changelist to a New CCR

Question:

Moving a cancelled changelist to a new CCR requires re-submitting content in a new changelist with the new CCR number. True or False?

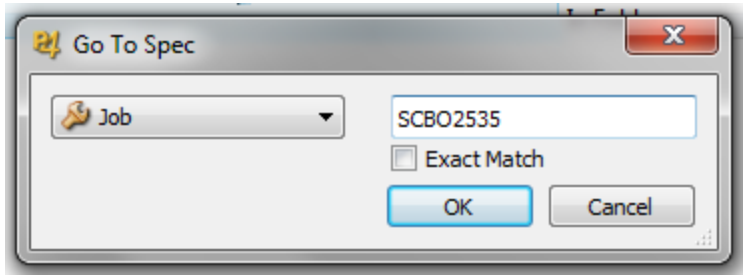
Answer: False.

You are merely updating the same changelist with the new job number. You are not re-submitting the changes.

Part 3: Back out Changes Using Perforce

Backout Changes using Perforce

- Only way backout some (rather than all) of the changelists submitted while In_BASE:
  1. Log into Perforce.
  2. Search > Go To…
     - Keyboard shortcut: Ctrl + G.
  3. Choose job.
  4. Enter your CCR number.
  5. Click OK.



Backout changes using Perforce (cont.)

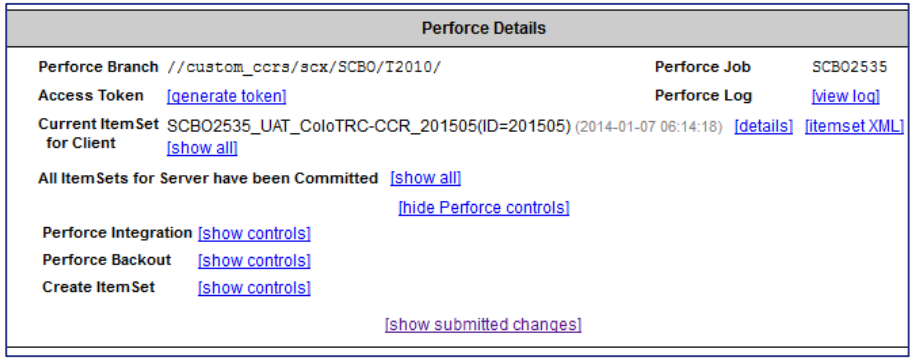
- Only way backout some (rather than all) of the changelists submitted while In_BASE:
  6. Right-click > Undo Changes in Changelist #####.
  7. Pending changelist: New.
  8. Click Save to Changelist.
  9. Repeat for each changelist that should be backed out.
     - Note: May need to submit changelist per instructions on next slide before backing out another changelist. Necessary when backing out multiple changelists on same file.

Backout changes using Perforce (cont.)

- Only way backout some (rather than all) of the changelists submitted while In_BASE:
  10. Select the Pending Change List Tab.
  11. Right-click changelist with the backout > Submit.
  12. Add in the changelist description.
  13. Enter job number (CCR number).
  14. Click Add.
  15. Click Submit.

Backout changes using Perforce (cont.)

- Only way backout some (rather than all of the changelists submitted while In_BASE:
  16. Open the CCR.
  17. Show Perforce controls.
  18. Click [show controls] for Create ItemSet.
  19. Select Target environment: BASE.
  20. Deploy ItemSet.



Quiz: Backing Out Changes

Question:

It is possible to backout changes from within the CCR application. True or False?

Answer: True.

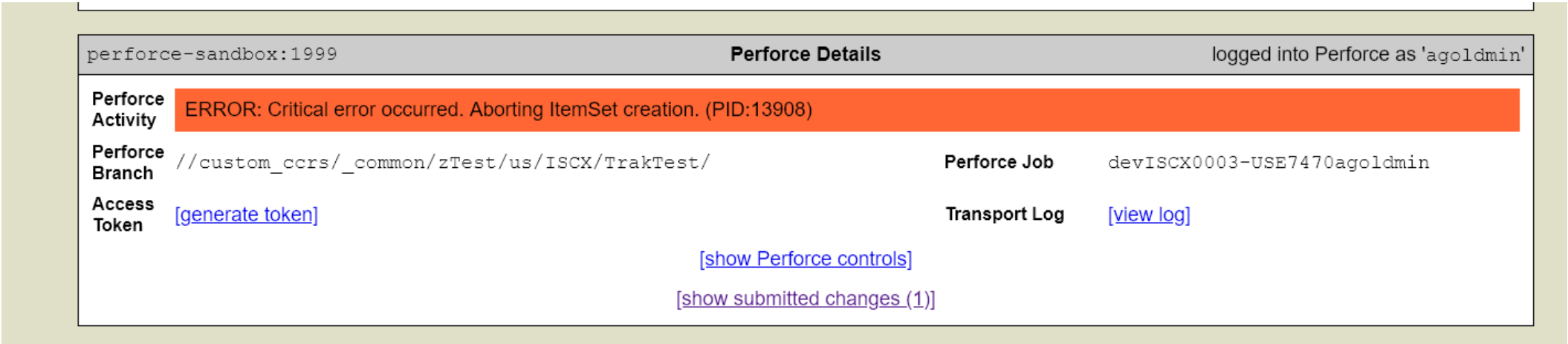
Frequently, changes can be backed out from within the CCR application when cancelling a CCR.

If it’s not possible, make sure to follow the previous steps to backout using Perforce.

Part 4: Resolving merge conflicts in P4V

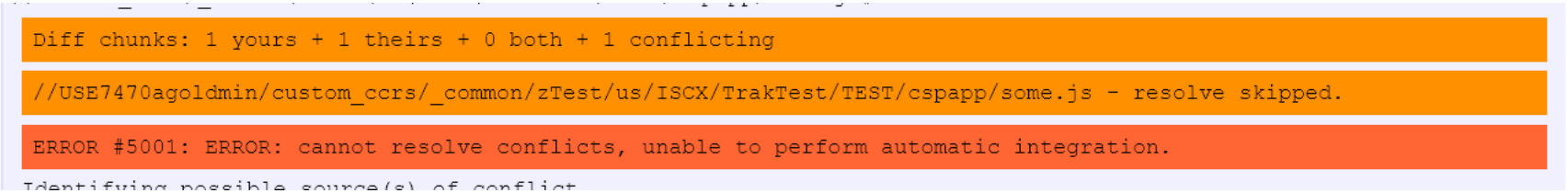
Resolving merge conflicts in P4V

- Sometimes you will see integration errors when moving a CCR to the next phase.



Resolving merge conflicts in P4V (cont.)

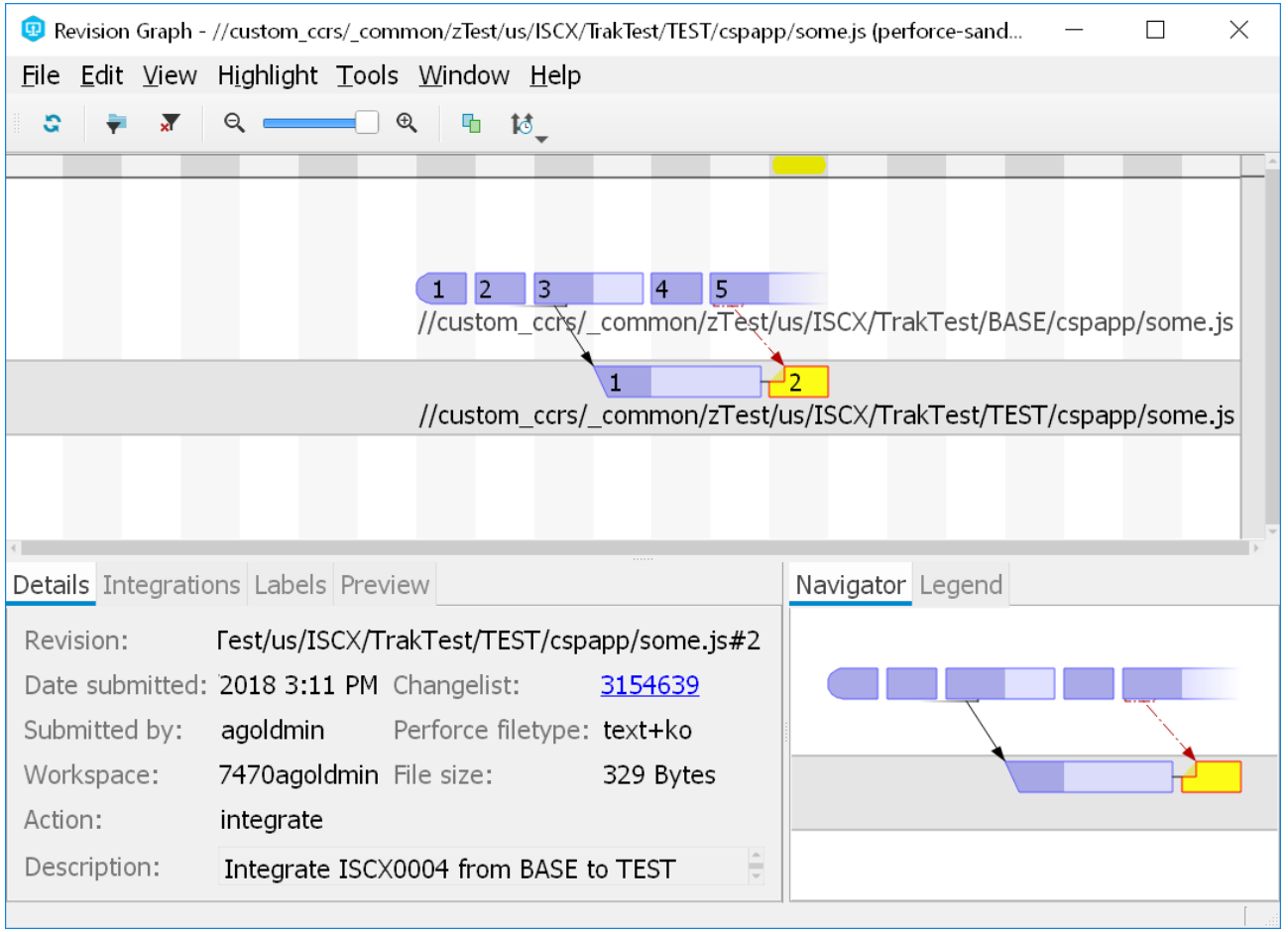
- You can check the Transport Log for details of the error.



- This will tell you both what the error was, and which file caused the problem.

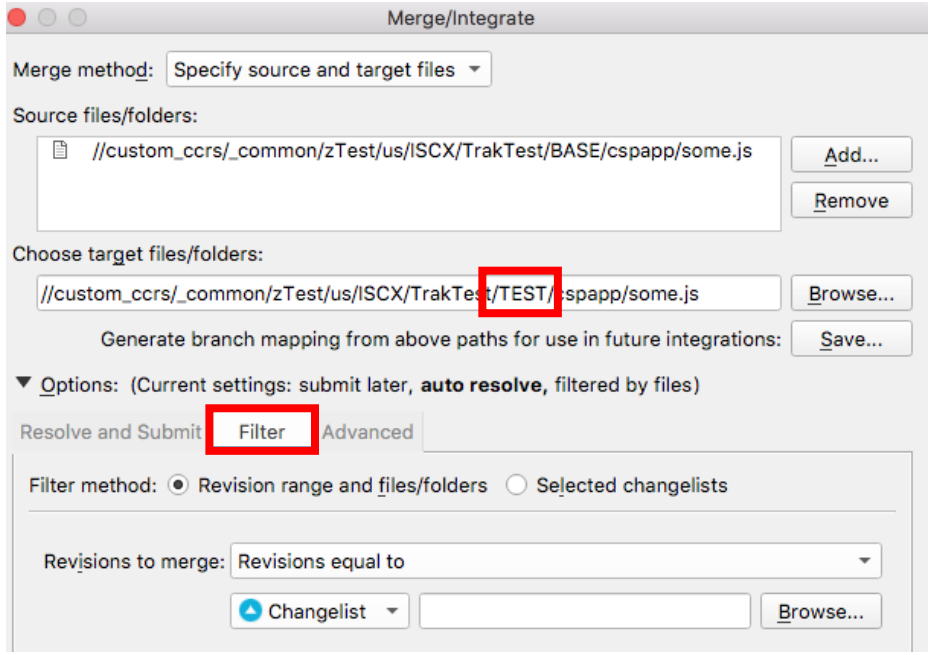
Resolving merge conflicts in P4V (cont.)

- Looking at the file's integration history can be helpful to understand why there are conflicts.
- In this example, revision 5 needed to progress without revision 4.
  - Screenshot shows the state after manual integration.



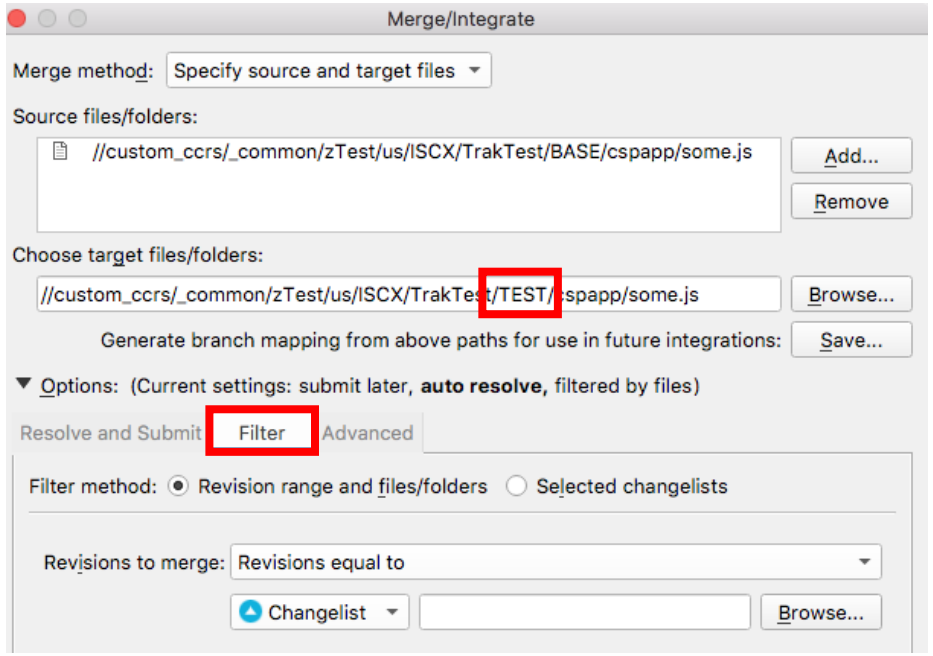
Resolving merge conflicts in P4V (cont.)

- Find the conflicting file(s) in question in the workspace view in P4V and select "Merge/Integrate" from the context menu.
  - NOTE – this manual step is only for conflicting files; after correcting them by hand use the CCR Integration Tool to manually integrate the remainder of the files.



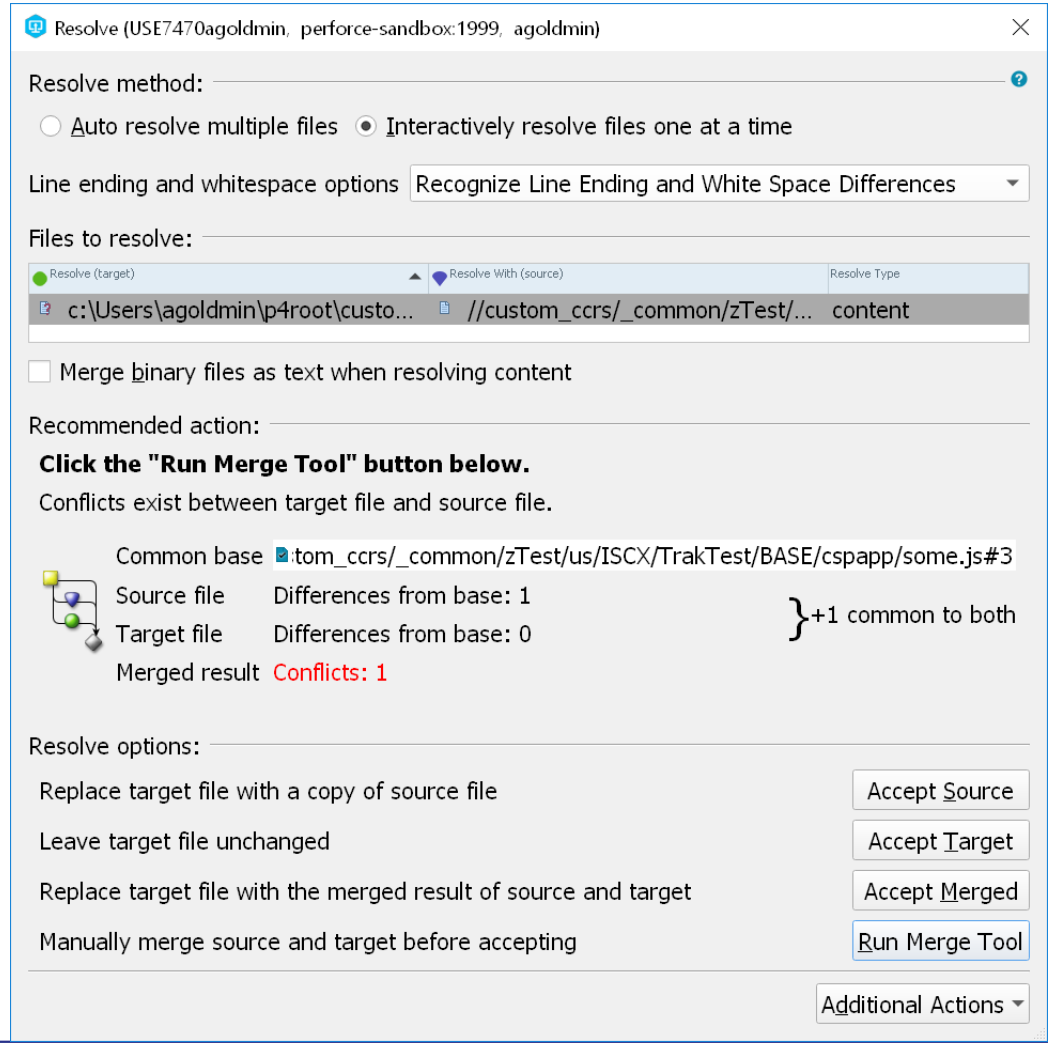
Resolving merge conflicts in P4V (cont.)

- Make sure the target path is set to the corresponding file in the TEST branch.
  - Target is *case sensitive* (i.e. 'TEST' not 'test').
- On "Filter" tab, select just the changelists you want to integrate.
- NOTE: After finishing the merge, the merged file(s) will need to be resolved before they can be submitted.



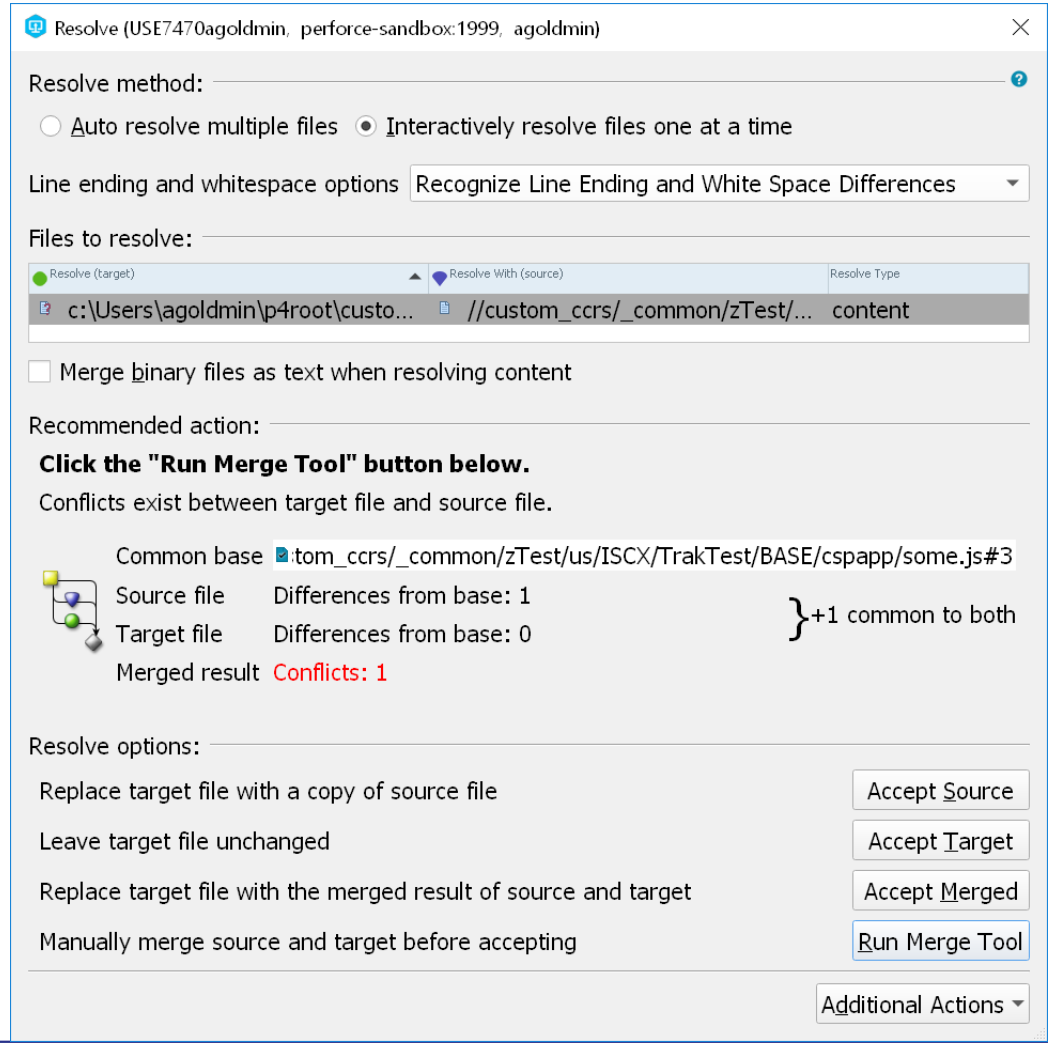
Resolving merge conflicts in P4V (cont.)

- Find the file in the target branch and select “Resolve” from the context menu.
- You will see a summary of the changes in the source and target files and options to resolve the merge.



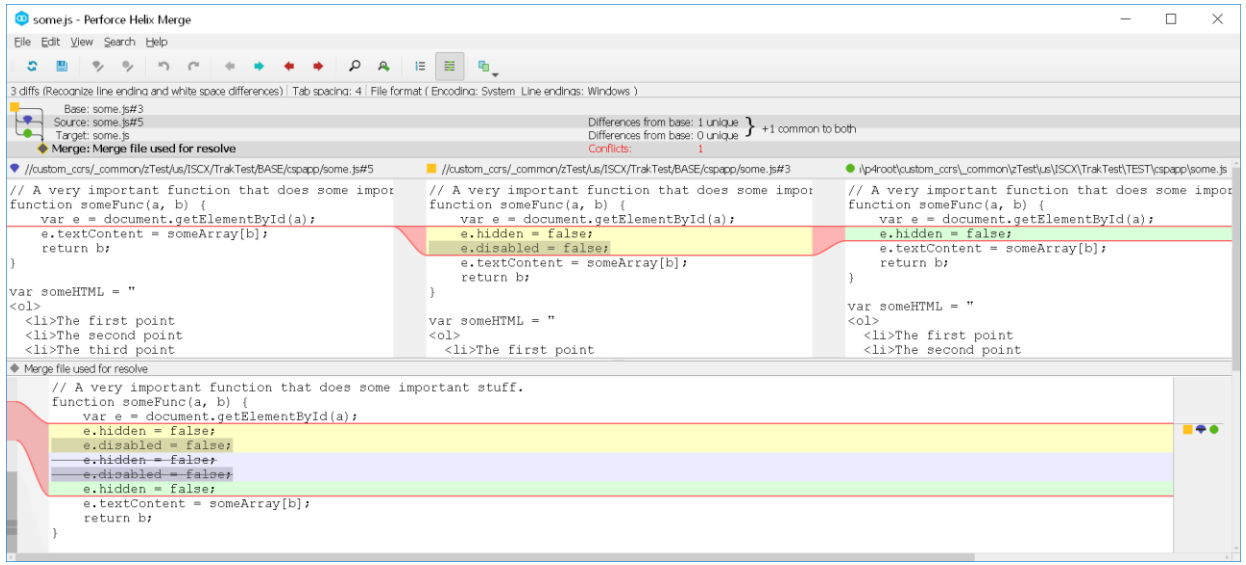
Resolving merge conflicts in P4V (cont.)

- Run the merge tool to see a view of the file contents in the various versions of the file so that you can decide how best to merge.
- This requires some knowledge of what the contents of the file mean to ensure that the result makes sense.



Resolving merge conflicts in P4V (cont.)

- The merge tool on the right shows a view of the file contents in the base, source, and target revisions, as well as the diffs between them.
- It is basically a diff view, but with three files rather than two.
- Select the icons to the right of the conflicting lines to select the desired version.
- NOTE: Edits can also be made to get the desired result.



Resolving merge conflicts in P4V (cont.)

- Click the “Save” icon once the merge is complete.
- When prompted about replacing the item in your workspace with the merged file, answer “Yes.”
- Now the change is ready to be submitted to Perforce.
  - **Don’t forget to attach the Perforce job (i.e the CCR ID) to the changelist so that CCR can see the changelist and include it in the ItemSet.**

Resolving merge conflicts in P4V (cont.)

- If there are other files that didn’t have a conflict, they can be auto-integrated with the CCR Perforce Tools.
- Make sure an ItemSet is created; deploy it and continue moving the CCR forward.

Quiz: Resolving Merge Conflicts

Question:

If the integration of one file in the CCR needs to be merged by hand, then all files in that CCR must be manually merged as well. True or False?

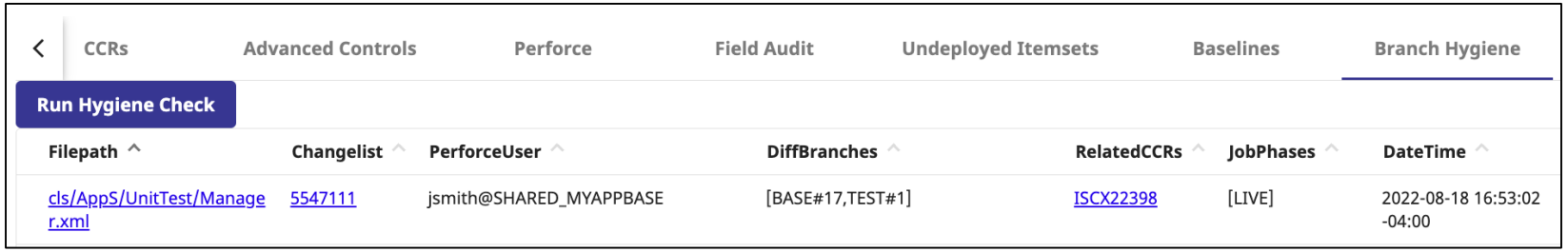
Answer: False.

Only the conflicting file must be manually merged; once the conflicting file is integrated, the other files can be integrated using the Perforce tools on the CCR.

Part 5:
Fixing integration history with p4 integrate -f

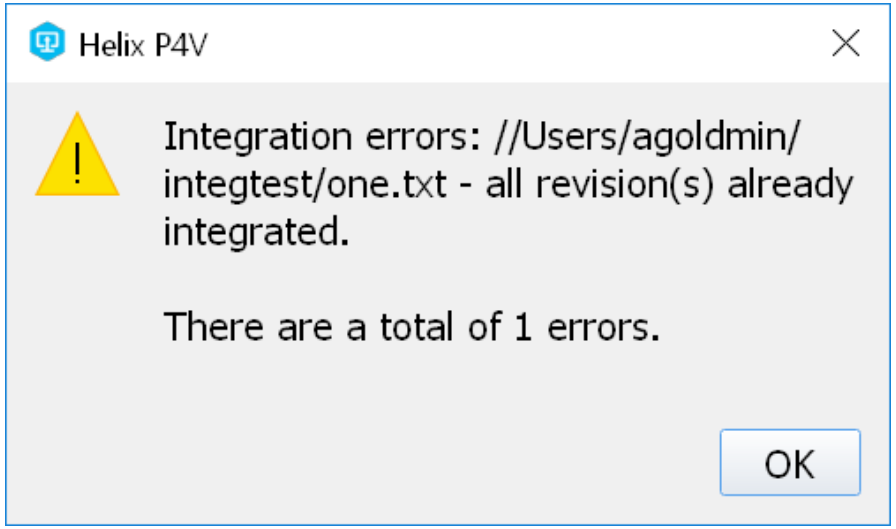
Branch Hygiene Tool

- Branch Hygiene tab on system definition shows files out of sync between branches.
  - Future changes to the files may have merge conflicts.
- To fix discrepancies:
  - For cancelled CCRs not backed out, back them out.
  - Use P4V to manually integrate changes.
    - Be sure to create a CCR and attach changelists.



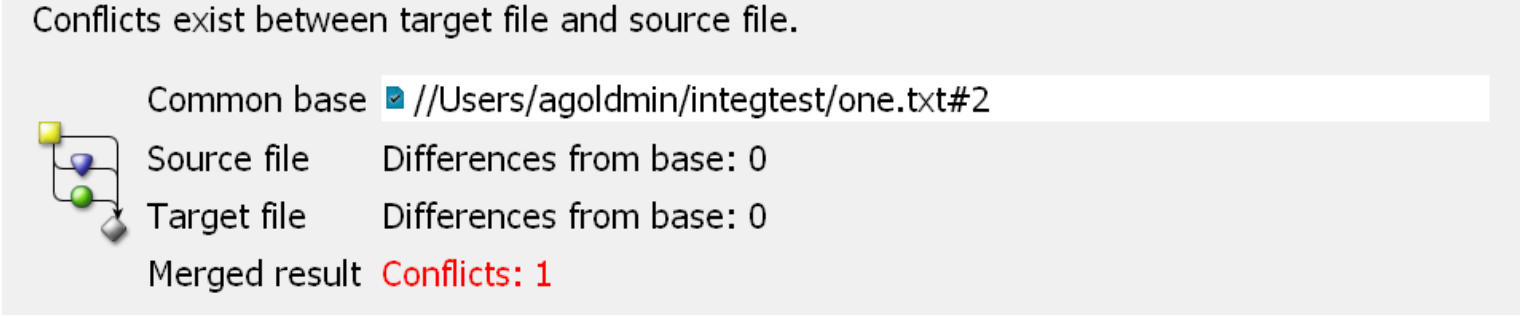
Fixing integration history with integrate -f

- Sometimes there are mistakes in p4 integrations.
  - E.g. resolving “accept target” instead of “accept source”.
  - Or integrating from the wrong source.



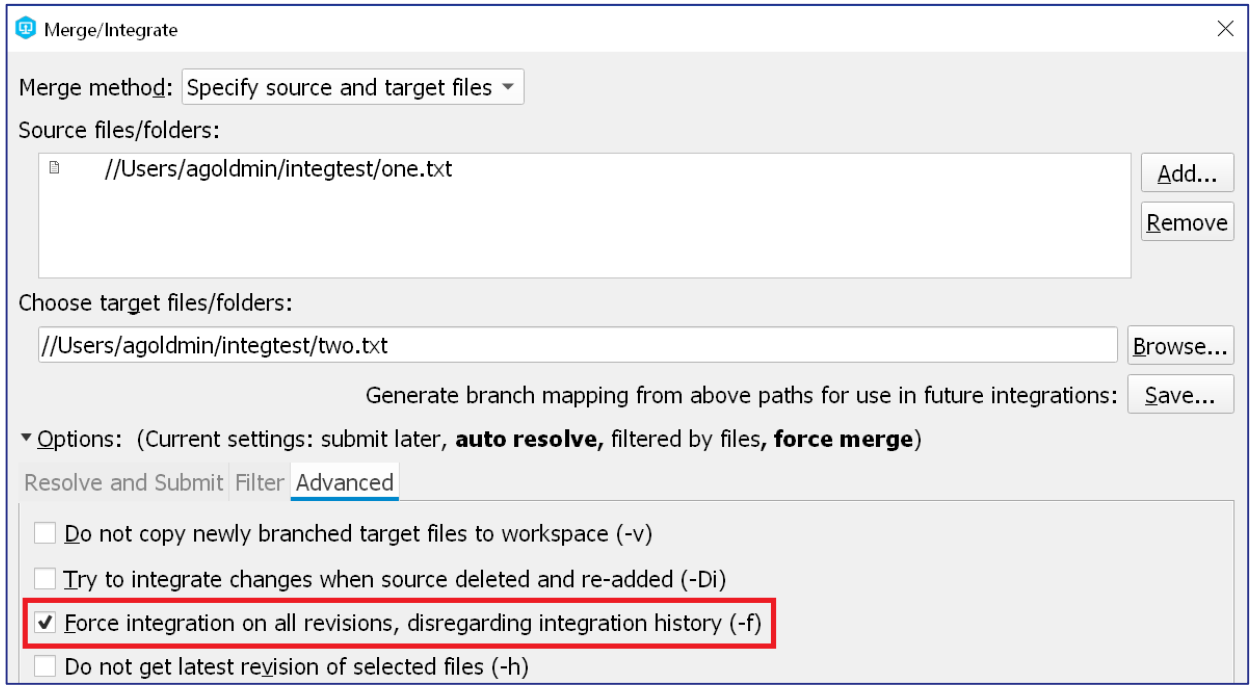
Fixing integration history with integrate -f (cont.)

- Perforce won’t let you integrate the same change twice into the same file.
- Possibly more changes were checked into the source file, causing merge conflicts.



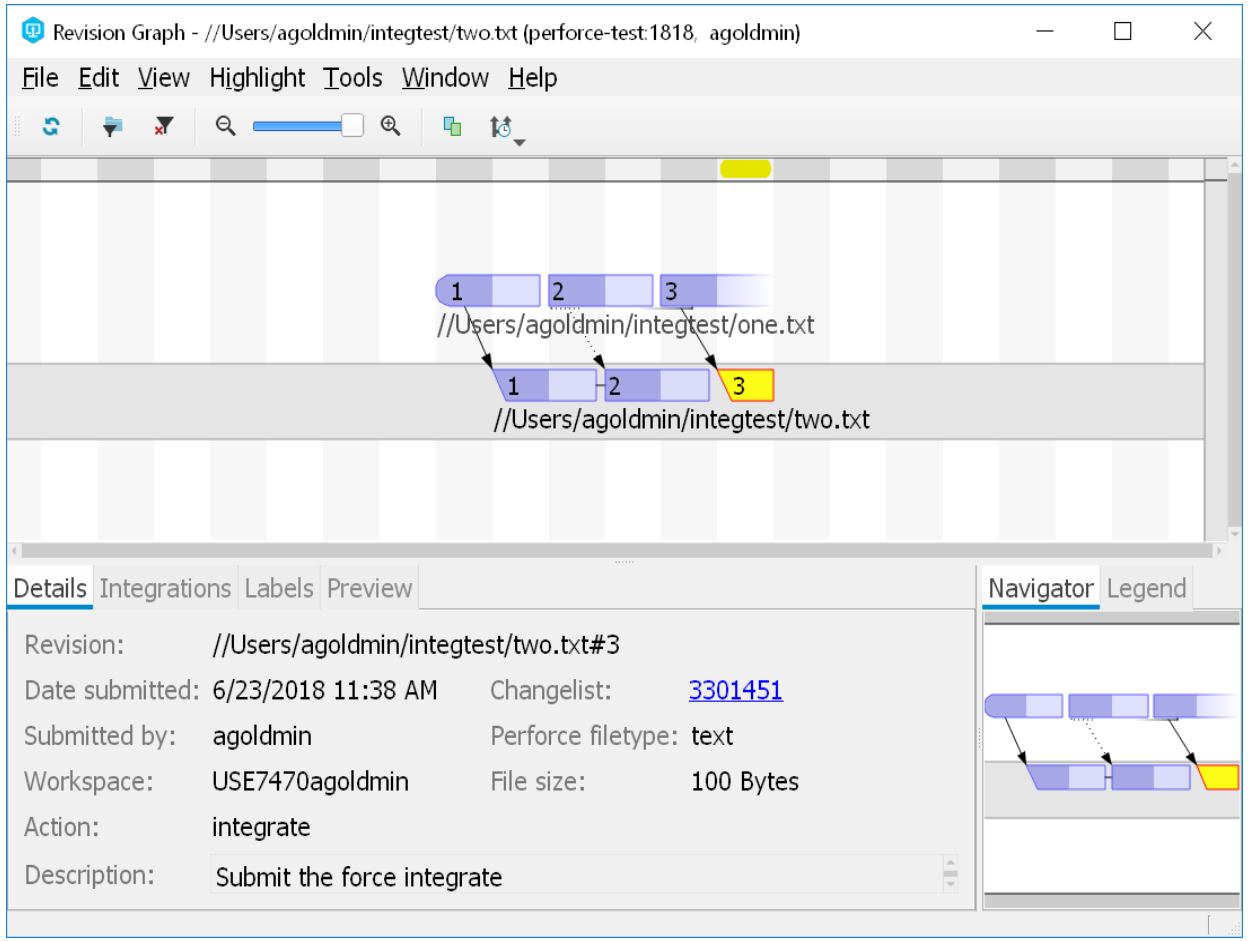
Fixing integration history with integrate -f (cont.)

- "p4 integrate -f" can help if you have a version of the file you want to copy forward to all branches.
- It will copy all the file contents and use it as a basis for all future integration history.



Fixing integration history with integrate -f (cont.)

- Results in Revision Graph like this:
- Note the diagonal corner on two.txt#3.
- The integration from one.txt#3 overwrites two.txt#2.



Part 6: Perforce Refresh

Perforce Refresh

- Refreshing the Perforce branches based on contents of LIVE:
  - To ensure that all configuration matches in each Perforce branch.
  - Clears integration history to prevent merge conflicts.
  - Can be done during working hours as it will have no impact on performance for users.
  - Before refresh ensure that:
    - Source Control Hooks (Tier 1) and/or Change Control Hooks (Tier 2) have been Locked.
    - Progress all CCR’s to terminal state (Closed, Merged, or Cancelled).

Perforce Refresh (cont.)

- Refreshing the Perforce branches based on contents of LIVE:
  - Perforce refresh done as part of rebaseline.
  - See rebaselining in ICC520 for details on how to perform refresh as part of rebaseline.

Quiz: Perforce Refresh

Question:

A Perforce refresh should be done during off hours because it can affect performance for the client. True or False?

Answer: False.

Summary

- What are the key points for this module?