IBM z Systems

Administrator Documentation

Josh Wisniewski, TPF Toolkit/Debugger Architect,

TPF Toolkit Technical Lead, TPF Development Lab

3/24/2015

Disclaimer

- Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

- Today...
- In a future release...
- When, where, webinar?

Administrator installation and enterprise configuration documentation today...

- The existing tutorials and documentation were written in a vague manner because TPF Toolkit is highly customizable and each customer's configuration may be unique. Further, critical configuration topics are often located in documentation that is not referenced by the tutorials making the information difficult to find. And expertise in IBM Installation Manager, Packaging Utility and so on, are assumed.
- As a result, we have received many questions, comments and PMRs on the 4.2 administrator installation and enterprise configuration update process.

Administrator documentation, in a future release...

- We are re-writing our administrator tutorials and documentation to help make these processes more clear. The intention of this effort is to
  - Create a script that administrators can follow during the installation and enterprise configuration update process
  - Provide extensive cross references that provide more detailed information
  - Provide more thorough examples
  - And even include screenshots.

When, where, webinar?

- These updates will be released in a future release of the TPF Toolkit (it is not being delivered in V.next).
- We are planning an administrator installation and enterprise configuration update webinar after the documentation update is released. We anticipate delivering this webinar in June 2015.
- We will notify our known contacts and post to the TPF Blog as more details become available.
- Please let me know if you'd like to be added to the TPF Toolkit email distribution.

Questions?

IBM z Systems

V06004S Save/Load project build list

Josh Wisniewski, TPF Toolkit/Debugger Architect,

TPF Toolkit Technical Lead, TPF Development Lab

3/24/2015

Disclaimer

- Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

- V06004S Save/Load project build list
- Demonstration of solution

V06004S Save/Load project build list

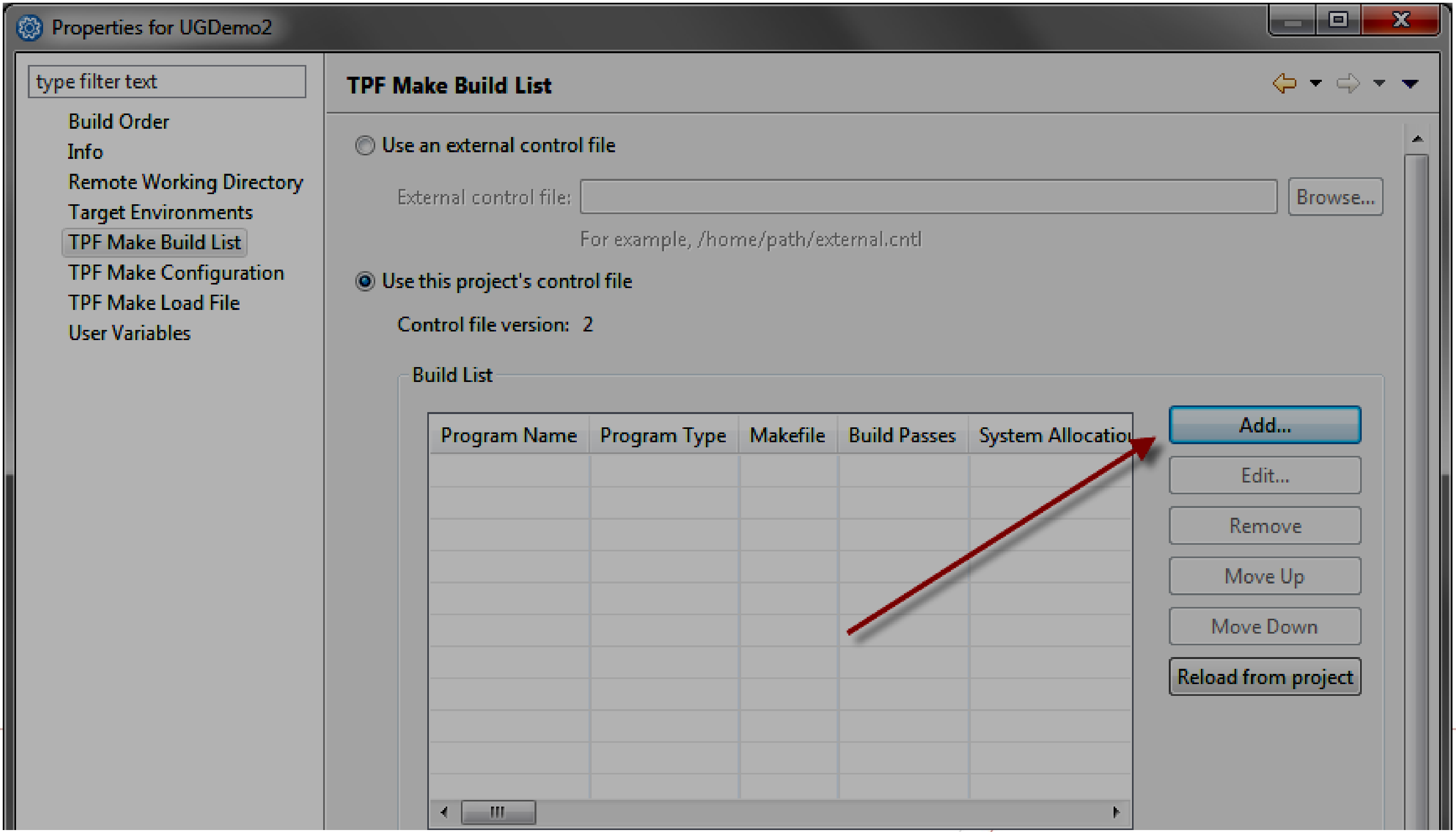
- This is an older requirement, probably opened against TPF 4.1 build solution.
- Abstract: A TPF Toolkit Project "Build List" is populated via a panel available in the project properties. An enhancement to save a build list and reload a saved build list would be useful.
- In this presentation, I will demonstrate how this requirement can be satisfied with the existing maketpf build solution. With this presentation, this requirement will be considered satisfied.

Demonstration of solution

- The z/TPF maketpf solution makes this easy to accomplish.
- We will use TPF Toolkit to create multiple maketpf.cntl files, which determine which modules will be built when you run the build action from a TPF Project.
- We will use TPF Toolkit to select which control file will be used for a TPF Project build.

Demonstration of solution

From the TPF Make Build List property page on the TPF Project, click Add to begin creating a maketpf.cntl file.



Properties for UGDemo2

type filter text

- Build Order
- Info
- Remote Working Directory
- Target Environments
- TPF Make Build List
- TPF Make Configuration
- TPF Make Load File
- User Variables

TPF Make Build List

○ Use an external control file

External control file: [ ] Browse...

For example, /home/path/external.cntl

● Use this project's control file

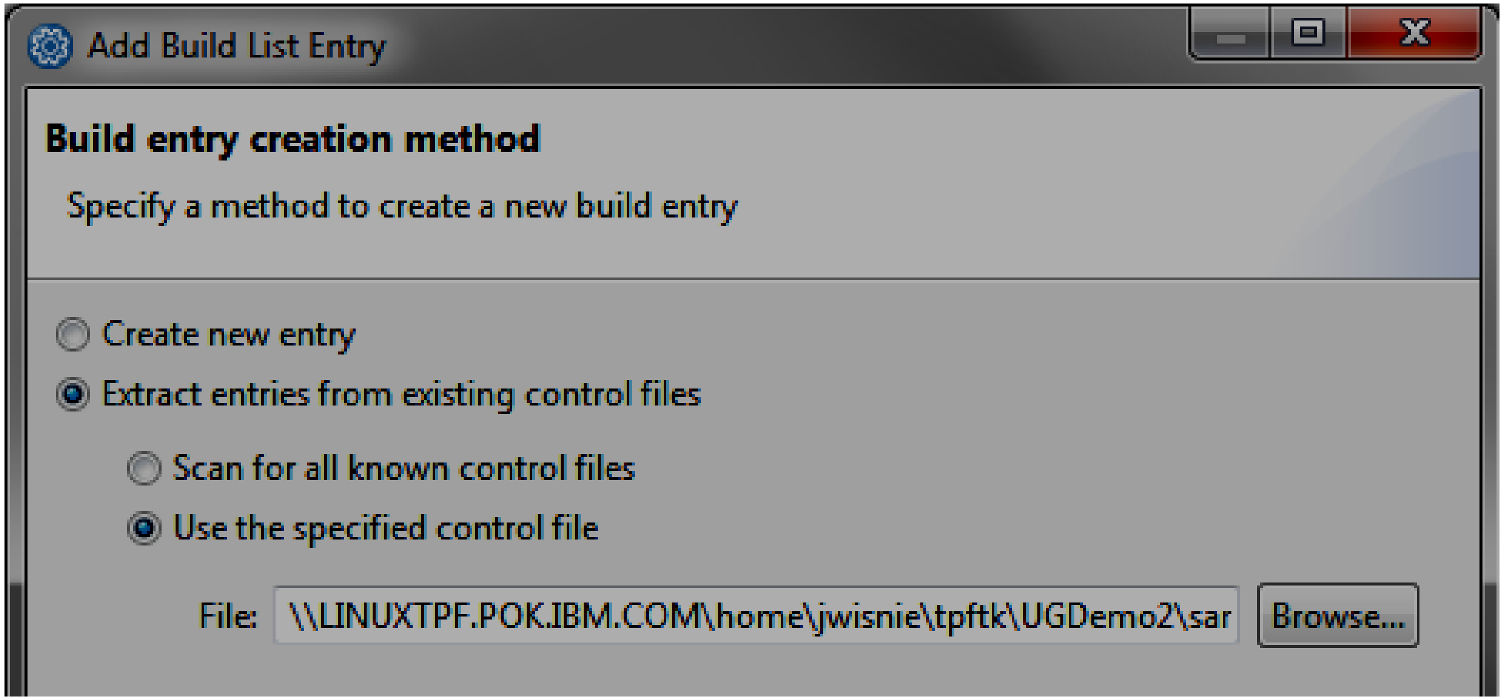
Control file version: 2

Build List

Program Name	Program Type	Makefile	Build Passes	System Allocatio

Demonstration of solution

Create new entries or extract entries from existing control files.



Build entry creation method

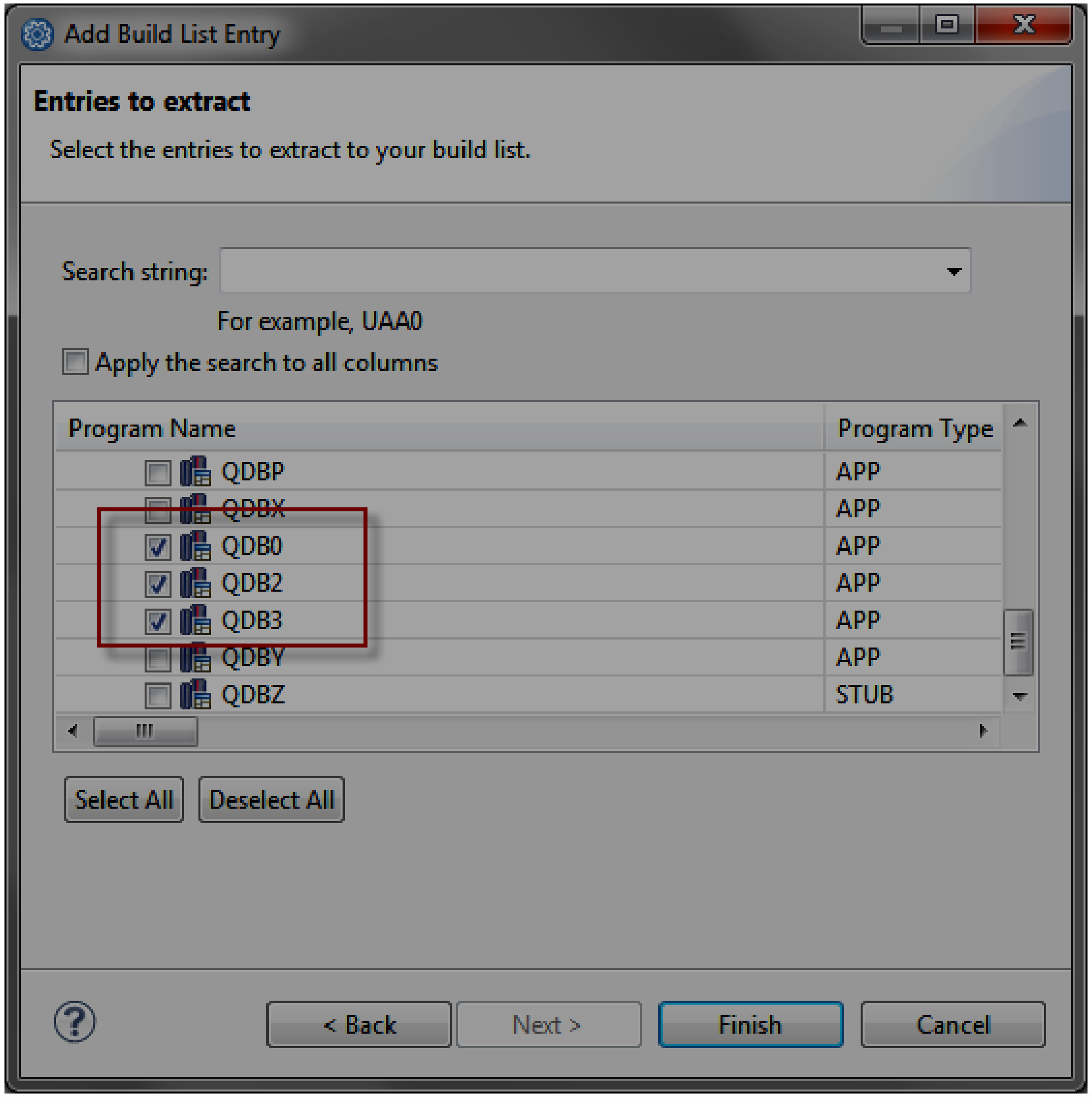
Specify a method to create a new build entry

- ○ Create new entry
- ● Extract entries from existing control files
  - ○ Scan for all known control files
  - ● Use the specified control file

File: \\LINUXTPF.POK.IBM.COM\home\jwisnie\tpftk\UGDemo2\sar

Demonstration of solution

If extracting, select the entries to use. Click finish.


Add Build List Entry

Entries to extract

Select the entries to extract to your build list.

Search string:

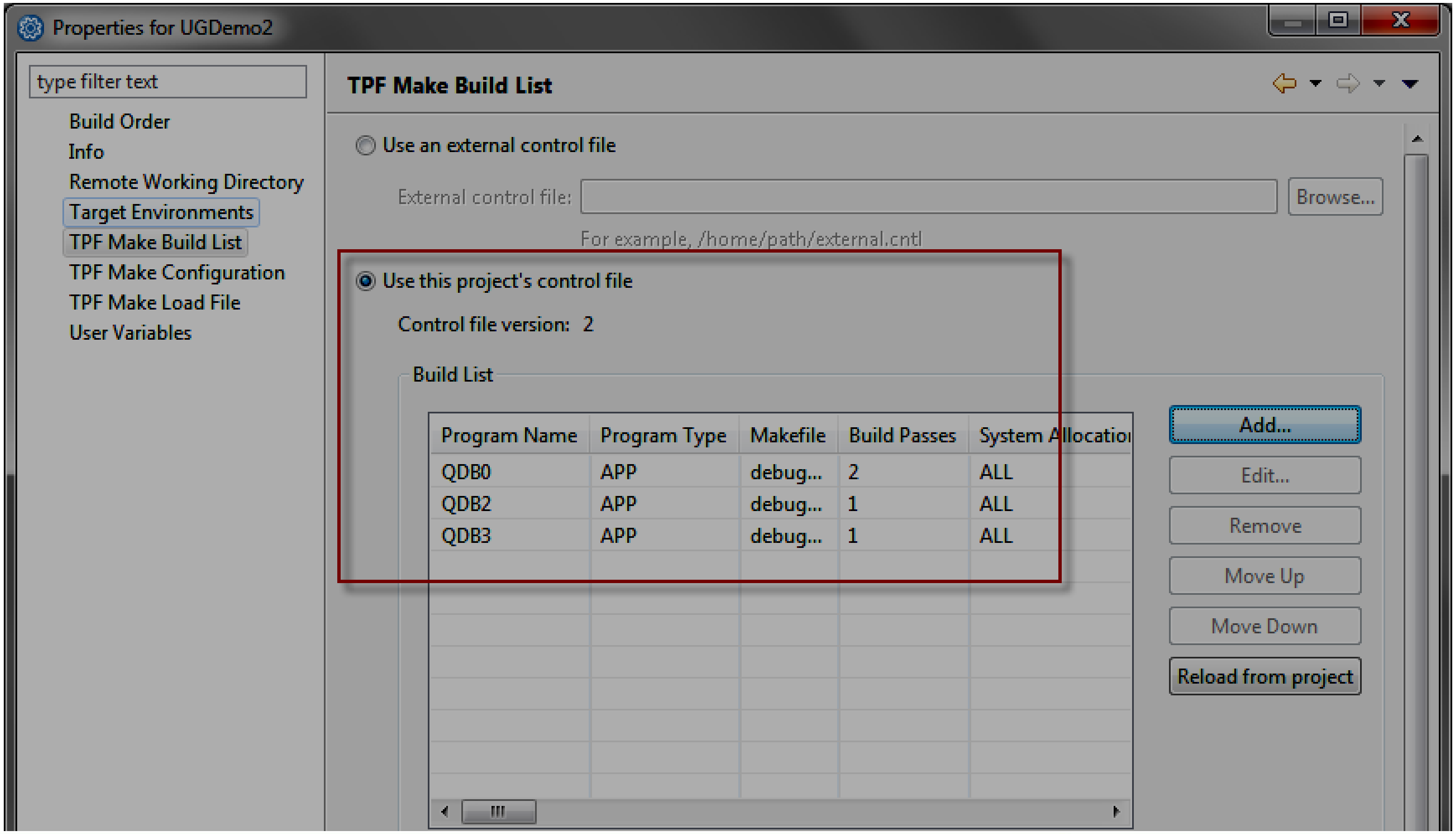
For example, UAA0

☐ Apply the search to all columns

Program Name	Program Type
☐ QDBP	APP
☐ QDBX	APP
✓ QDB0	APP
✓ QDB2	APP
✓ QDB3	APP
☐ QDBY	APP
☐ QDBZ	STUB

Demonstration of solution

A project control file now exists.



Properties for UGDemo2

type filter text

- Build Order
- Info
- Remote Working Directory
- Target Environments
- TPF Make Build List
- TPF Make Configuration
- TPF Make Load File
- User Variables

TPF Make Build List

Use an external control file

External control file:

For example, /home/path/external.cntl

[✓ Use this project's control file]

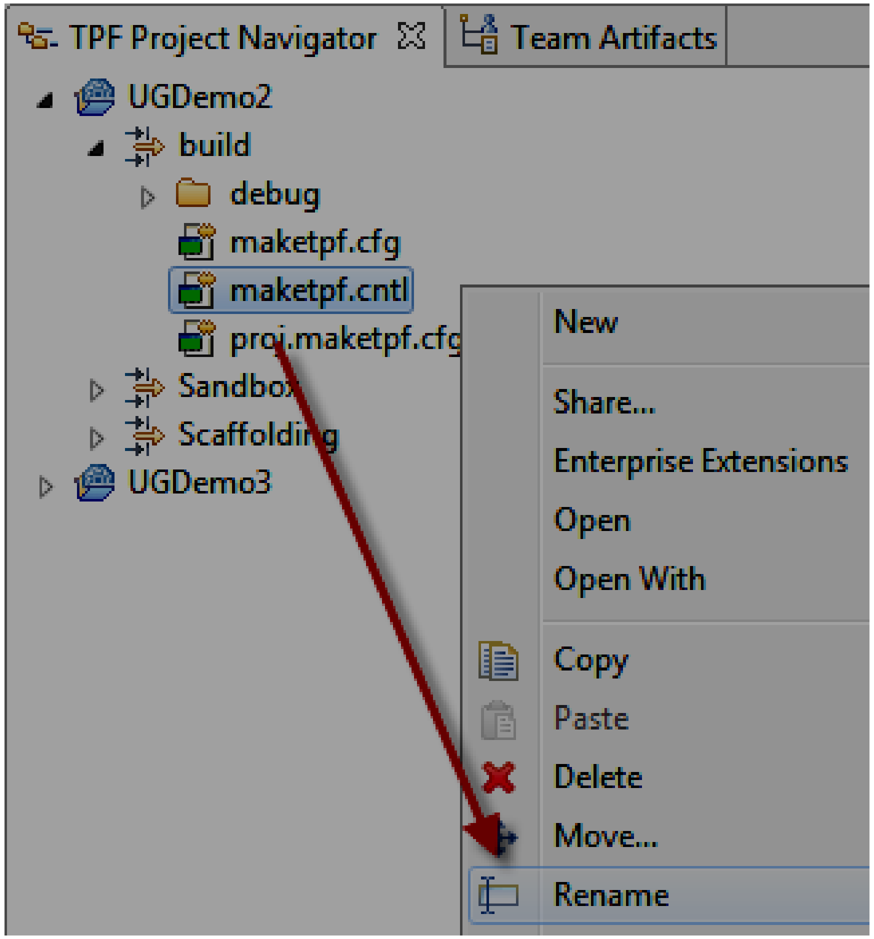
Control file version: 2

Build List

Program Name	Program Type	Makefile	Build Passes	System Allocatio
QDB0	APP	debug...	2	ALL
QDB2	APP	debug...	1	ALL
QDB3	APP	debug...	1	ALL

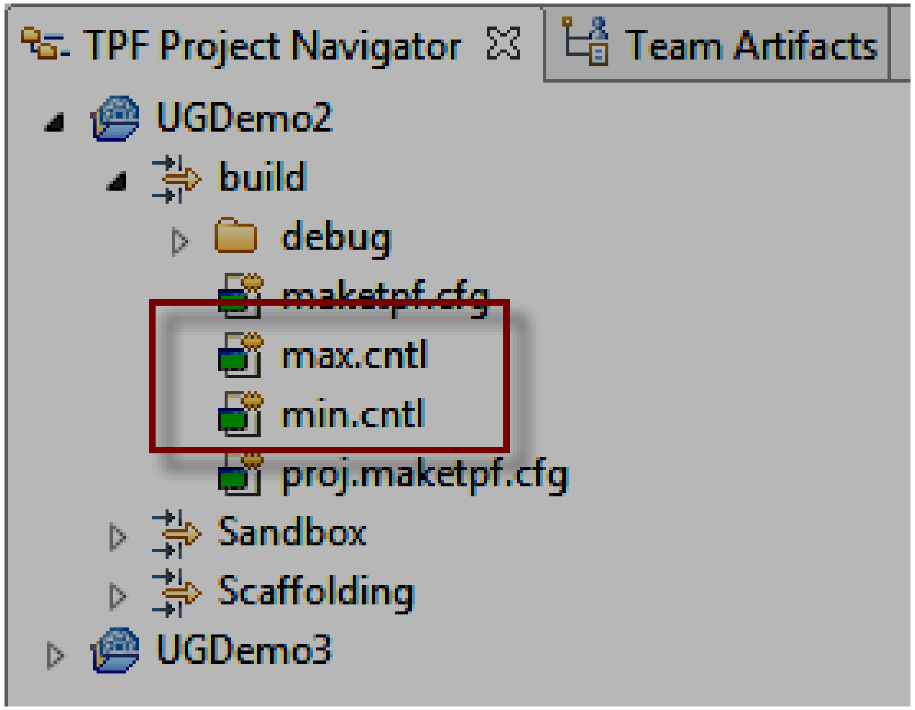
Demonstration of solution

Locate the project control file and rename it.



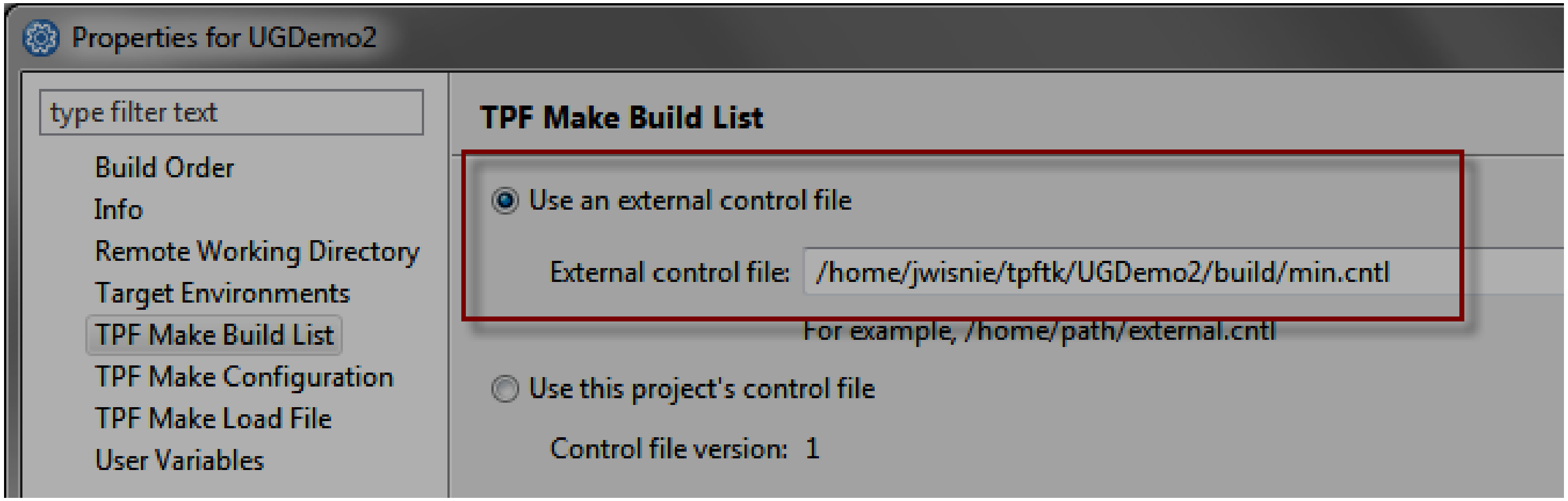
Demonstration of solution

Repeat the above procedure to create as many maketpf control files as needed.



Demonstration of solution

Edit the build list properties again. Choose to use an external control file and locate the desired maketpf control that was previously created.



Properties for UGDemo2

type filter text

- Build Order
- Info
- Remote Working Directory
- Target Environments
- TPF Make Build List
- TPF Make Configuration
- TPF Make Load File
- User Variables

TPF Make Build List

(●) Use an external control file

External control file: /home/jwisnie/tpftk/UGDemo2/build/min.cntl

For example, /home/path/external.cntl

( ) Use this project's control file

Control file version: 1

If this way, you can create, manage, and use multiple build lists (maketpf control files) as needed.

Questions?

IBM z Systems

Rational Team Concert (RTC) Integration Feature Tips

Josh Wisniewski, TPF Toolkit/Debugger Architect,

TPF Toolkit Technical Lead, TPF Development Lab

3/24/2015

Disclaimer

- Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

- What is RTC?
- Link to webinar
- Organizing your code in RTC
- Defining a component and adding files
- Creating new files
- Auto check-in
- Suggested Build set up
- Suggested Build set up: End user example

What is RTC?

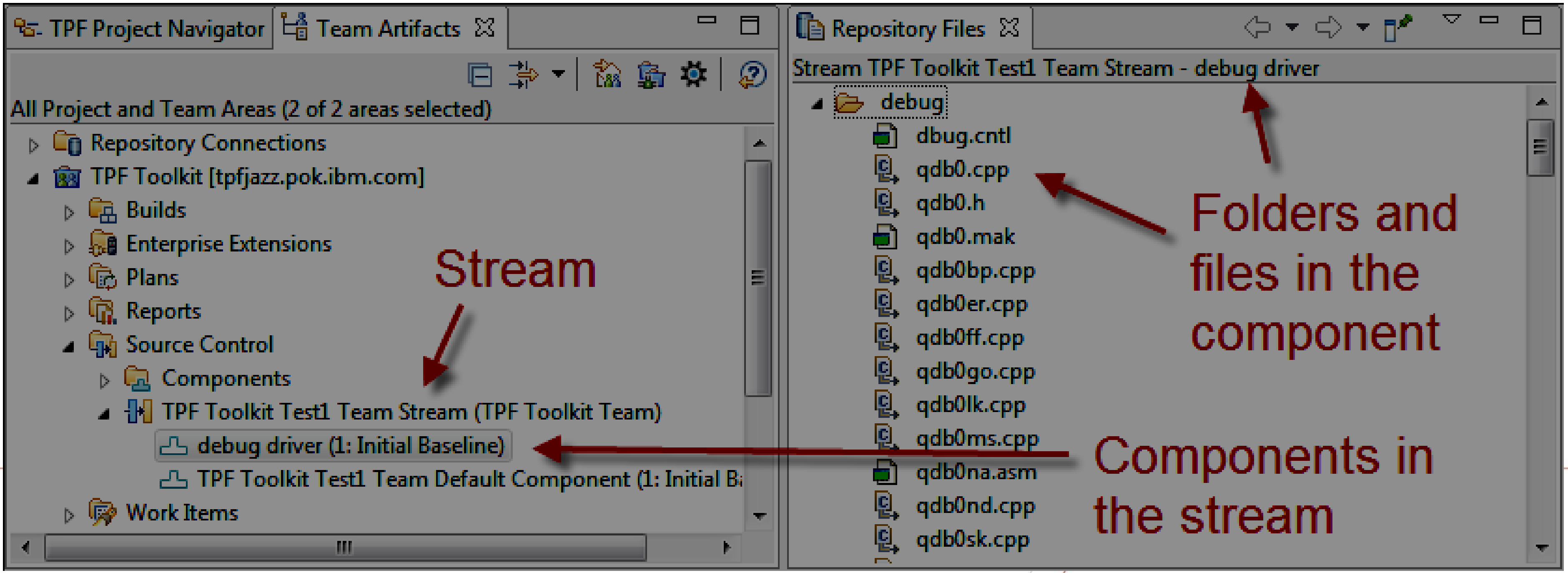
- RTC is an IBM software lifecycle tool. It provides source code management (SCM), defect tracking, planning, build and much more.
- TPF Toolkit RTC Integration feature is an optional component you can install into TPF Toolkit with the RTC client. It provides wizards, actions and such that link TPF Toolkit projects and RTC constructs for the TPF development environment.

Link to webinar

- https://www.ibm.com/developerworks/community/blogs/zTPF/entry/recording_of_introduction_to_rational_team_concert_and_tpf_toolkit_integration?lang=en
- Or search for “TPF toolkit RTC webinar”
- Topics discussed:
  - Installation of RTC integration feature
  - RTC concepts overview
  - RTC integration feature overview

Organizing your code in RTC

- In RTC terminology, your administrator defines a stream that contains components. Your administrator or users define components that contain packages, folders, files and etc.
- As such, your administrator needs to break your existing code base up into logical loadable working components.
- As we will see in this presentation, component definitions play a significant role in code development and TPF Projects.



All Project and Team Areas (2 of 2 areas selected)

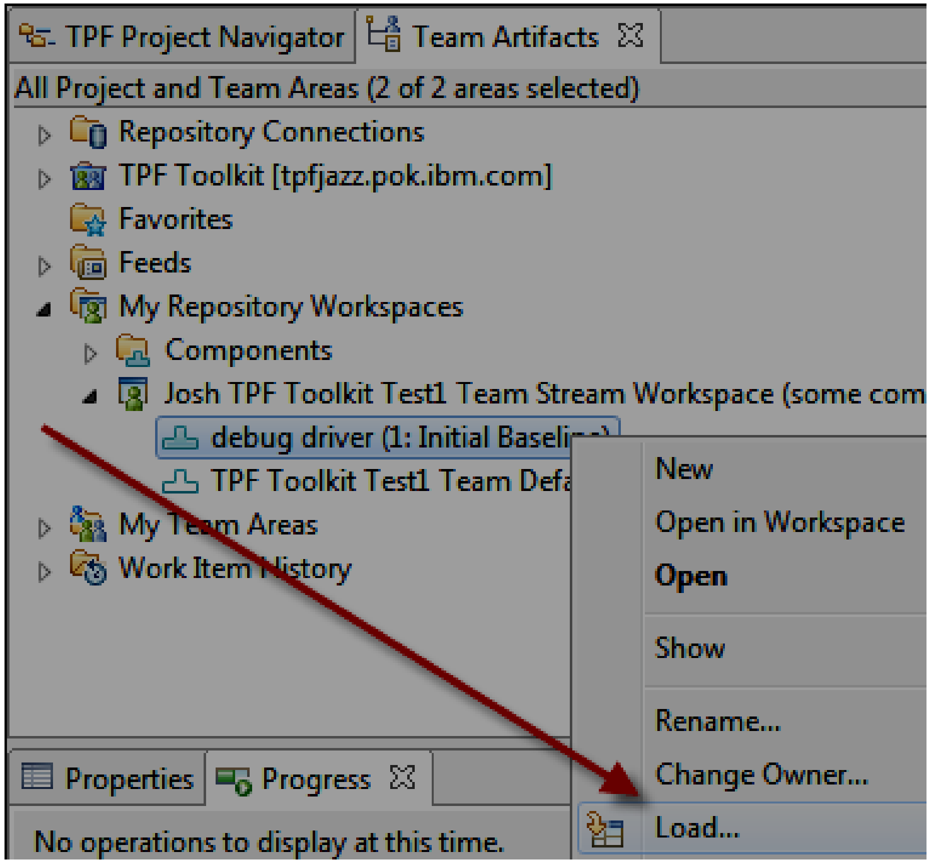
- Repository Connections
- TPF Toolkit [tpfjazz.pok.ibm.com]
  - Builds
  - Enterprise Extensions
  - Plans
  - Reports
  - Source Control
    - Components
    - TPF Toolkit Test1 Team Stream (TPF Toolkit Team)
      - debug driver (1: Initial Baseline)
      - TPF Toolkit Test1 Team Default Component (1: Initial Ba
  - Work Items

Stream TPF Toolkit Test1 Team Stream - debug driver

- debug
  - dbug.cntl
  - qdb0.cpp
  - qdb0.h
  - qdb0.mak
  - qdb0bp.cpp
  - qdb0er.cpp
  - qdb0ff.cpp
  - qdb0go.cpp
  - qdb0lk.cpp
  - qdb0ms.cpp
  - qdb0na.asm
  - qdb0nd.cpp
  - qdb0sk.cpp

Organizing your code in RTC

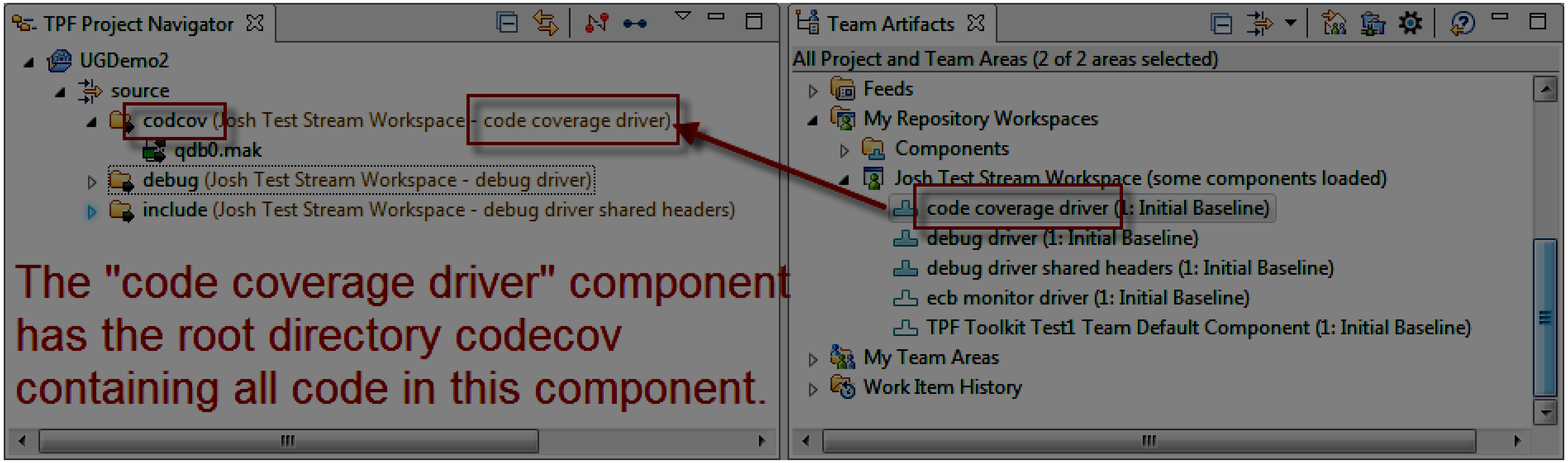
- The RTC team recommends that a component should have less than 100 files.
- If your administrator would like some guidance breaking up your code base, please contact us and we'll guide you to individuals who can help.



TPF Project Navigator
Team Artifacts
All Project and Team Areas (2 of 2 areas selected)
Repository Connections
TPF Toolkit [tpfjazz.pok.ibm.com]
Favorites
Feeds
My Repository Workspaces
Components
Josh TPF Toolkit Test1 Team Stream Workspace (some com
debug driver (1: Initial Baseli
TPF Toolkit Test1 Team Defa
My Team Areas
Work Item History
New
Open in Workspace
Open
Show
Rename...
Change Owner...
Load...
Properties
Progress
No operations to display at this time.

Defining a component and adding files

- When you create a new component, create a folder that will contain all of your source files for that component. Undesirable behavior will result if you do not have root folder for your component.


TPF Project Navigator

UGDemo2
- source
  - codcov (Josh Test Stream Workspace - code coverage driver)
    - qdb0.mak
  - debug (Josh Test Stream Workspace - debug driver)
  - include (Josh Test Stream Workspace - debug driver shared headers)

The "code coverage driver" component has the root directory codecov containing all code in this component.

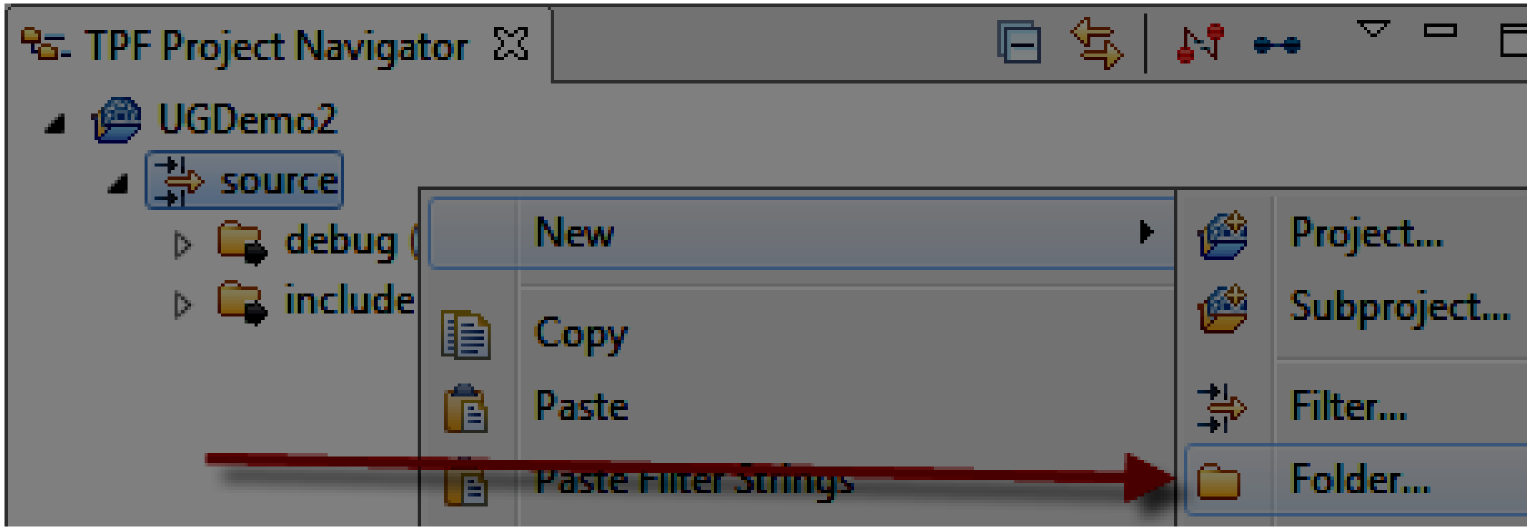
Team Artifacts

All Project and Team Areas (2 of 2 areas selected)
- Feeds
- My Repository Workspaces
  - Components
  - Josh Test Stream Workspace (some components loaded)
    - code coverage driver (1: Initial Baseline)
    - debug driver (1: Initial Baseline)
    - debug driver shared headers (1: Initial Baseline)
    - ecb monitor driver (1: Initial Baseline)
    - TPF Toolkit Test1 Team Default Component (1: Initial Baseline)
- My Team Areas
- Work Item History

Defining a component and adding files

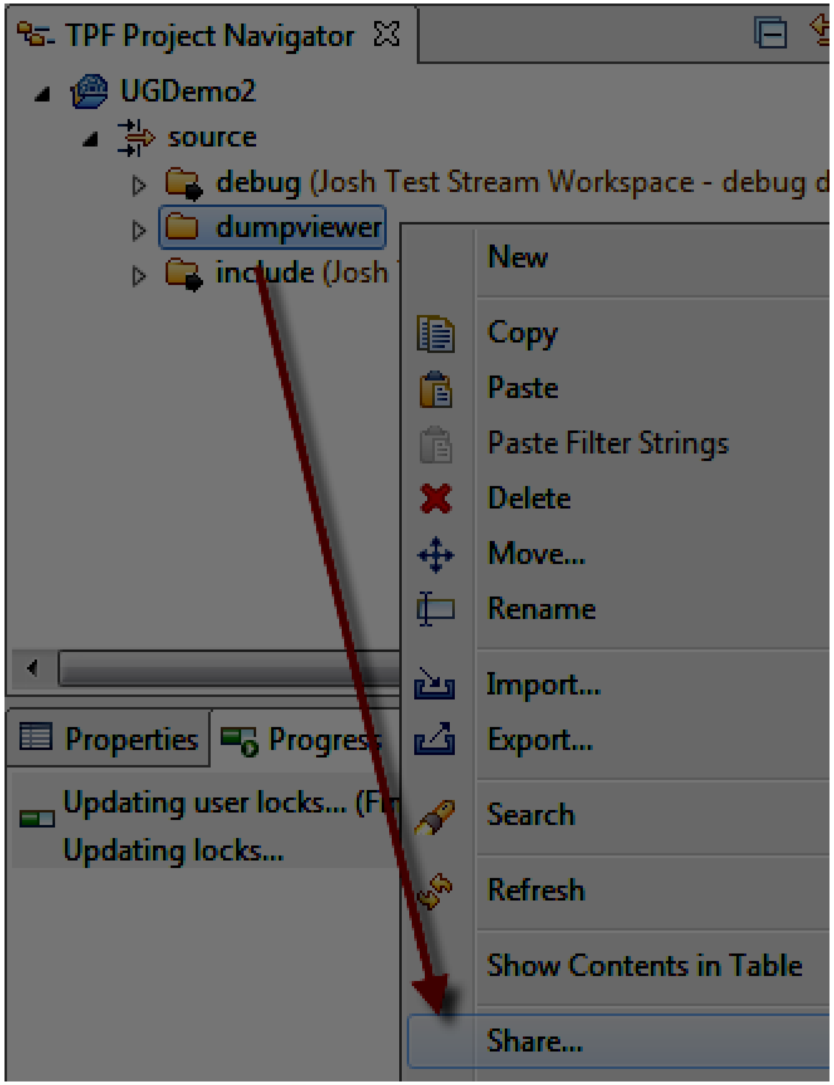
When defining a new component, use the following procedure:

In the TPF Project Navigator view in a TPF Project, create a new folder. Files can be created, pasted, and etc. under that folder now or later...



Defining a component and adding files

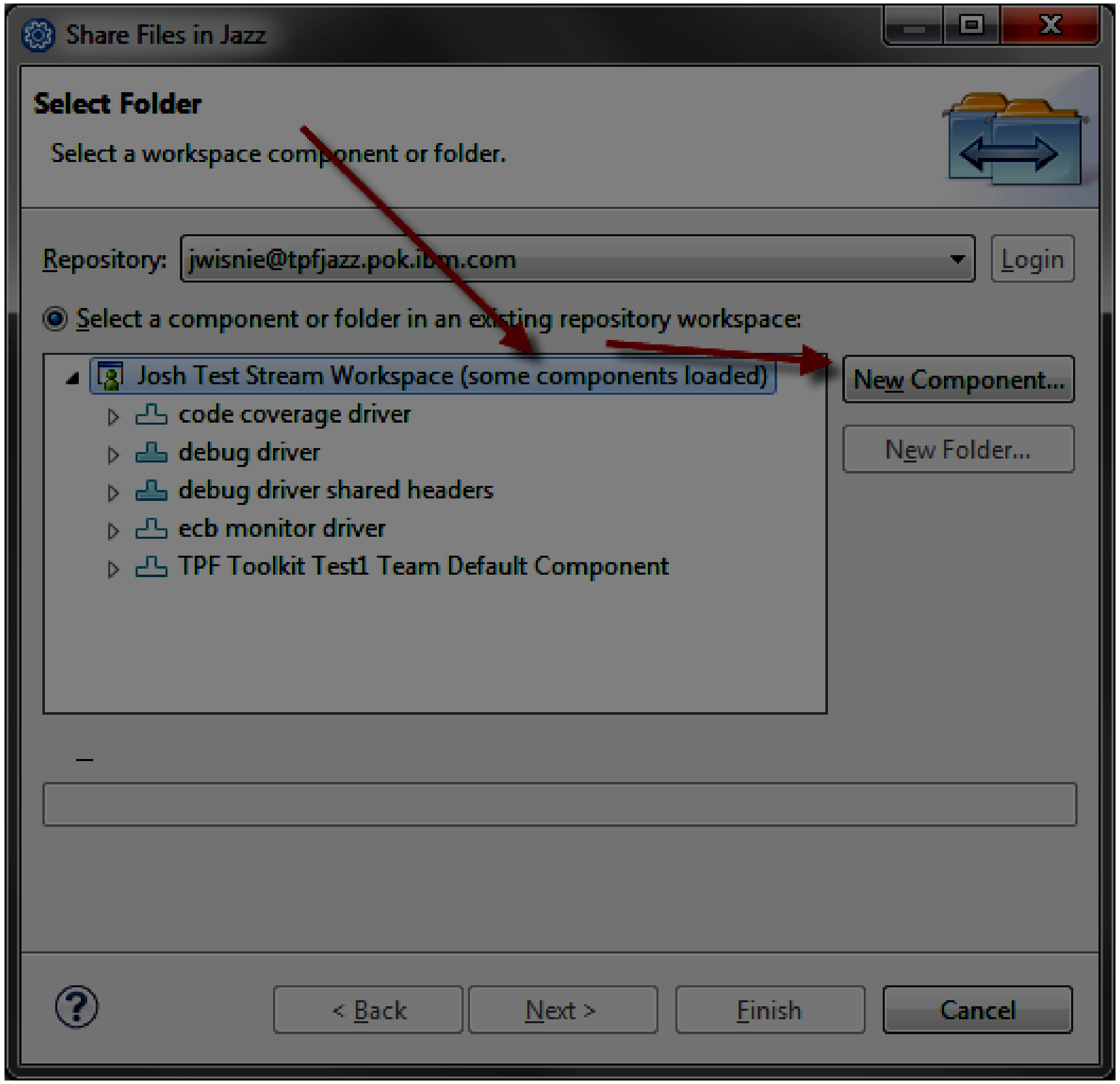
Right click the new folder and choose share. Share is the equivalent of creating a folder or file in a component (including check in).



Updating user locks... (F
Updating locks...

Defining a component and adding files

Choose the workspace and then the New Component button.



Share Files in Jazz

Select Folder

Select a workspace component or folder.

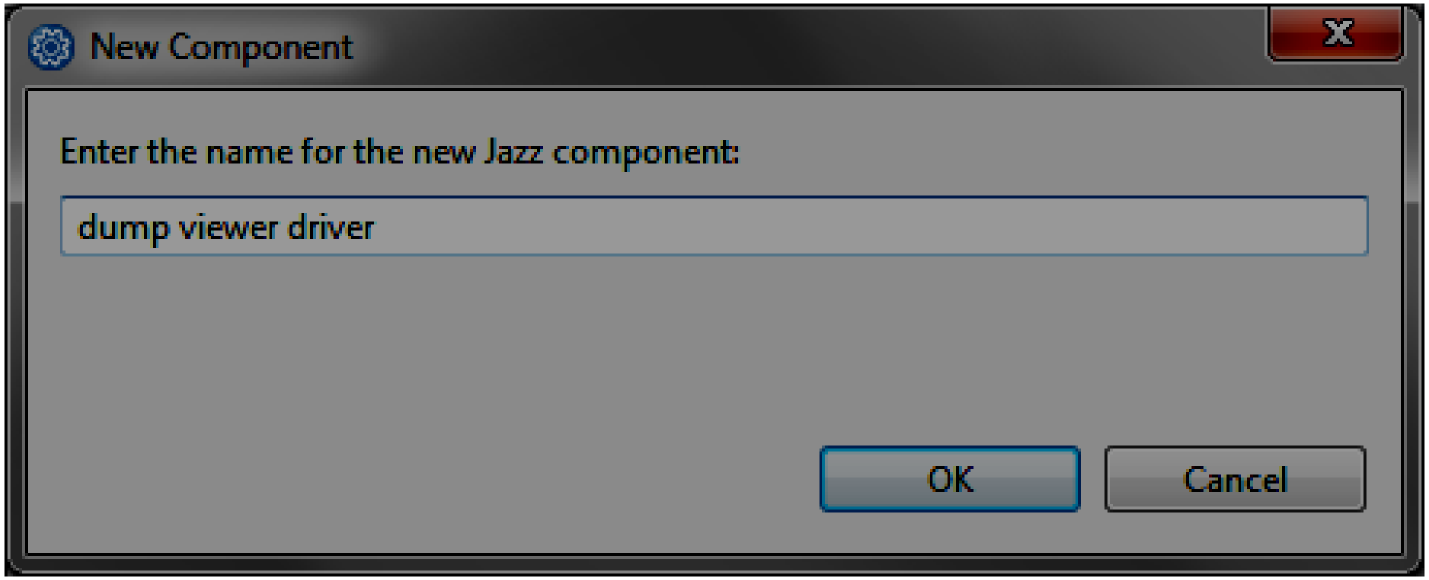
Repository: jwisnie@tpfjazz.pok.ibm.com

(●) Select a component or folder in an existing repository workspace:

- Josh Test Stream Workspace (some components loaded)
  - code coverage driver
  - debug driver
  - debug driver shared headers
  - ecb monitor driver
  - TPF Toolkit Test1 Team Default Component

Defining a component and adding files

Name your New Component and choose OK.

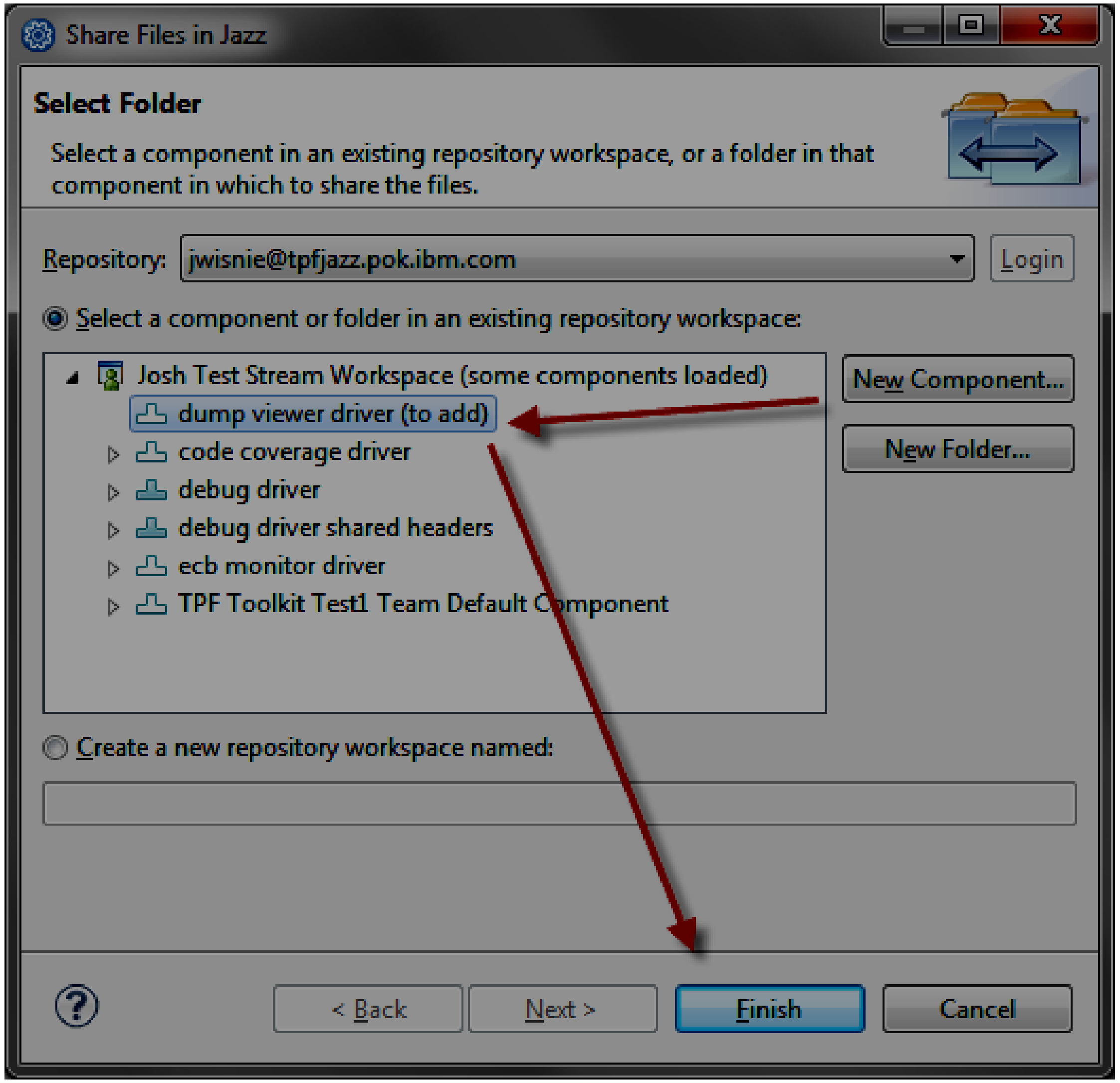


Enter the name for the new Jazz component:

dump viewer driver

Defining a component and adding files

Select the new component and choose finish.


Share Files in Jazz

Select Folder

Select a component in an existing repository workspace, or a folder in that component in which to share the files.

Repository: jwisnie@tpfjazz.pok.ibm.com

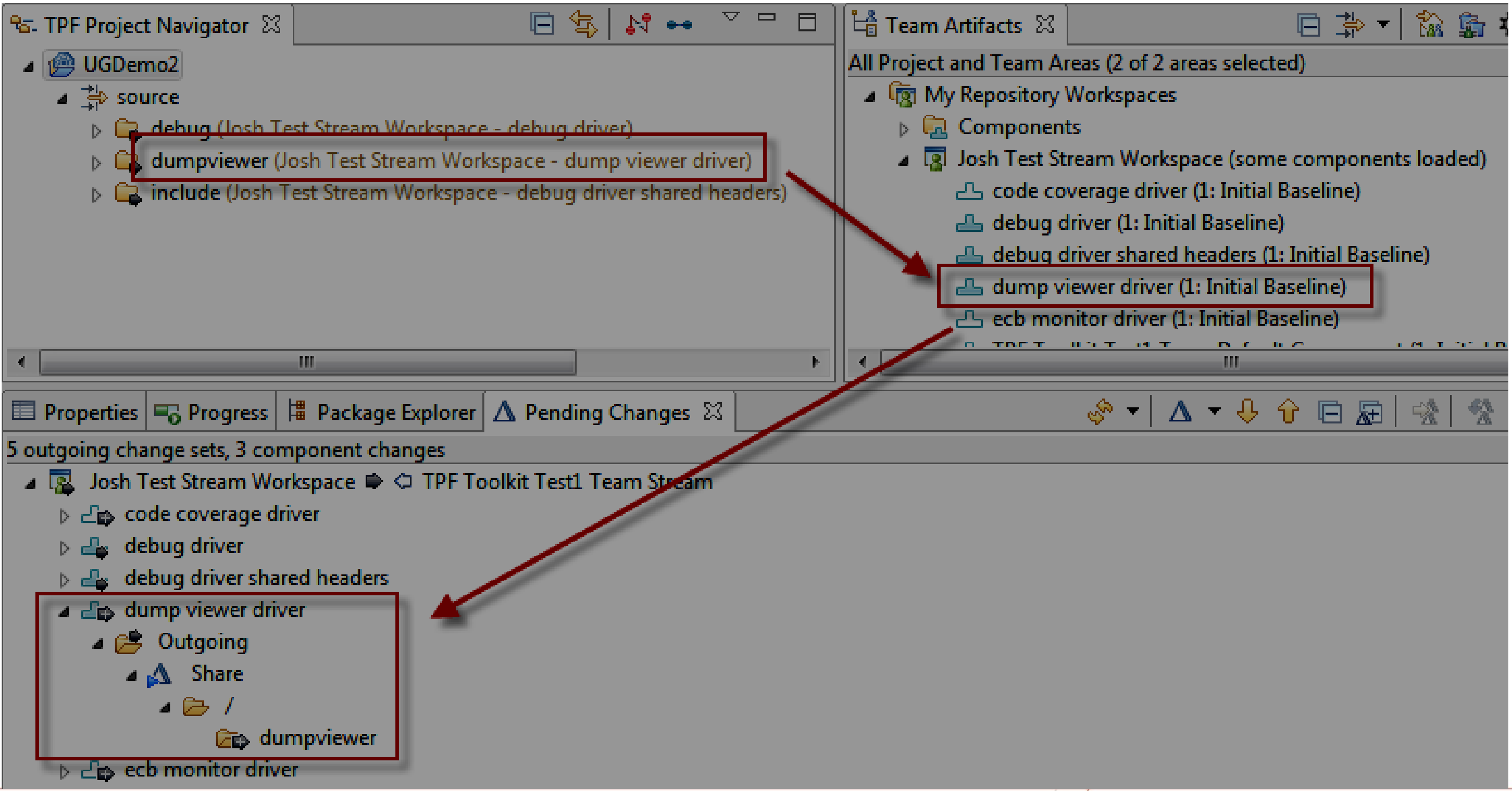
[✓ Select a component or folder in an existing repository workspace:]

- Josh Test Stream Workspace (some components loaded)
  - dump viewer driver (to add)
  - code coverage driver
  - debug driver
  - debug driver shared headers
  - ecb monitor driver
  - TPF Toolkit Test1 Team Default Component

Create a new repository workspace named:

Defining a component and adding files

Pending Changes view shows the new directory as checked in.



TPF Project Navigator

- UGDemo2
  - source
    - debug (Josh Test Stream Workspace - debug driver)
    - dumpviewer (Josh Test Stream Workspace - dump viewer driver)
    - include (Josh Test Stream Workspace - debug driver shared headers)

Team Artifacts

All Project and Team Areas (2 of 2 areas selected)

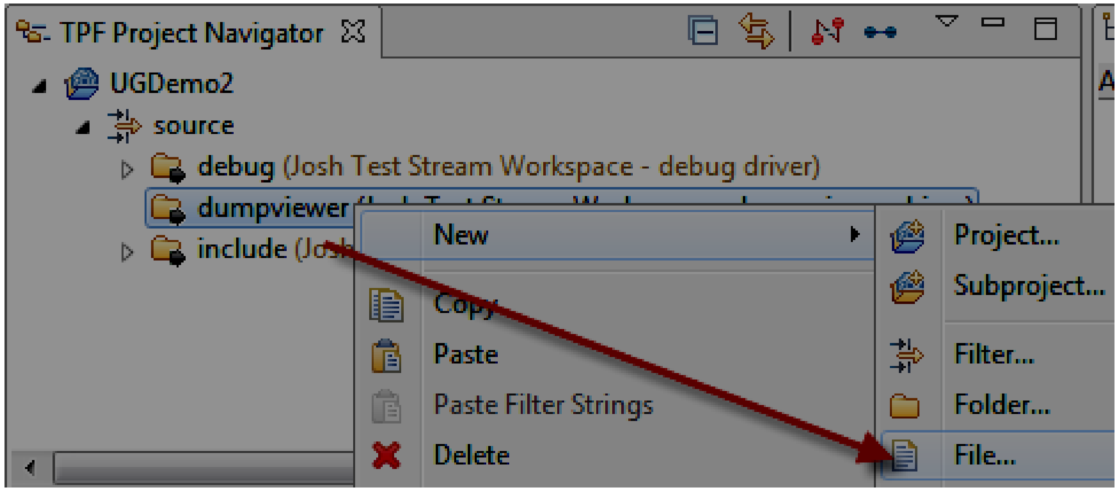
- My Repository Workspaces
  - Components
  - Josh Test Stream Workspace (some components loaded)
    - code coverage driver (1: Initial Baseline)
    - debug driver (1: Initial Baseline)
    - debug driver shared headers (1: Initial Baseline)
    - dump viewer driver (1: Initial Baseline)
    - ecb monitor driver (1: Initial Baseline)

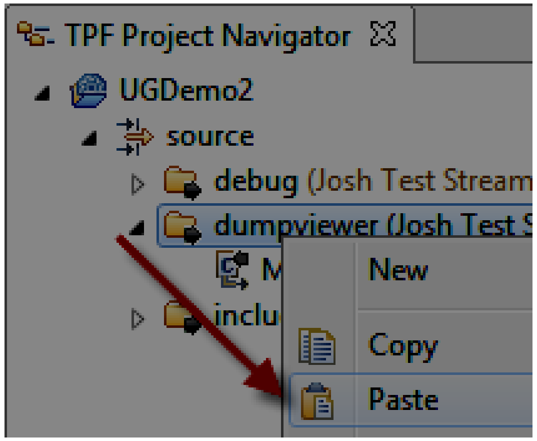
5 outgoing change sets, 3 component changes

- Josh Test Stream Workspace ➡ TPF Toolkit Test1 Team Stream
  - code coverage driver
  - debug driver
  - debug driver shared headers
  - dump viewer driver
    - Outgoing
      - Share
        - /
          - dumpviewer
  - ecb monitor driver

Defining a component and adding files

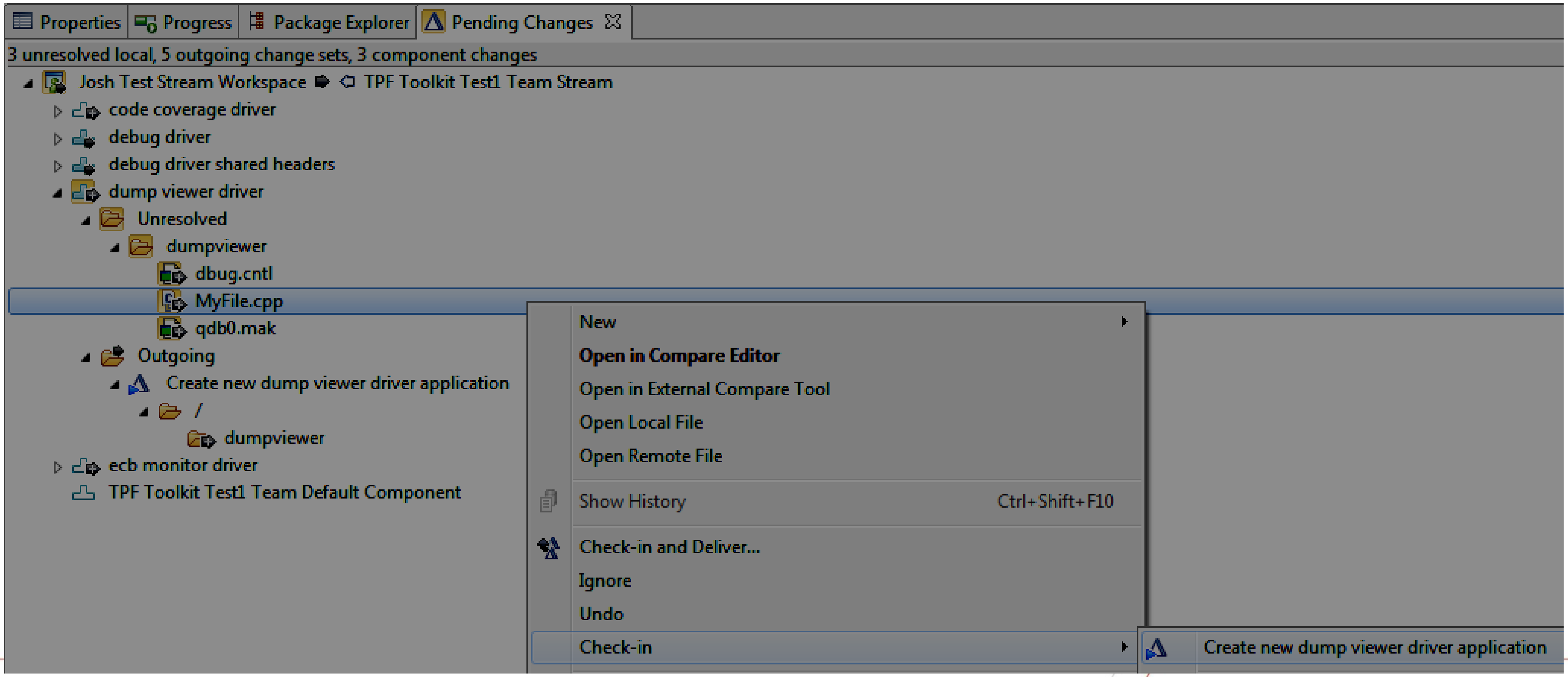
Create, copy/paste, and etc. folders and files under the component's root directory.





Defining a component and adding files

Now you need to check in your changes. There are a few ways to do this such as Refresh Sandboxes and Remote Changes action in the Pending Changes view, right clicking the unresolved files, choosing check-in and the desired change set...



3 unresolved local, 5 outgoing change sets, 3 component changes

```
Josh Test Stream Workspace ⇒ TPF Toolkit Test1 Team Stream
  code coverage driver
  debug driver
  debug driver shared headers
  dump viewer driver
    Unresolved
      dumpviewer
        dbug.cntl
        MyFile.cpp
        qdb0.mak
    Outgoing
      Create new dump viewer driver application
        /
          dumpviewer
  ecb monitor driver
  TPF Toolkit Test1 Team Default Component
```

New
Open in Compare Editor
Open in External Compare Tool
Open Local File
Open Remote File
Show History Ctrl+Shift+F10
Check-in and Deliver...
Ignore
Undo
Check-in
Create new dump viewer driver application

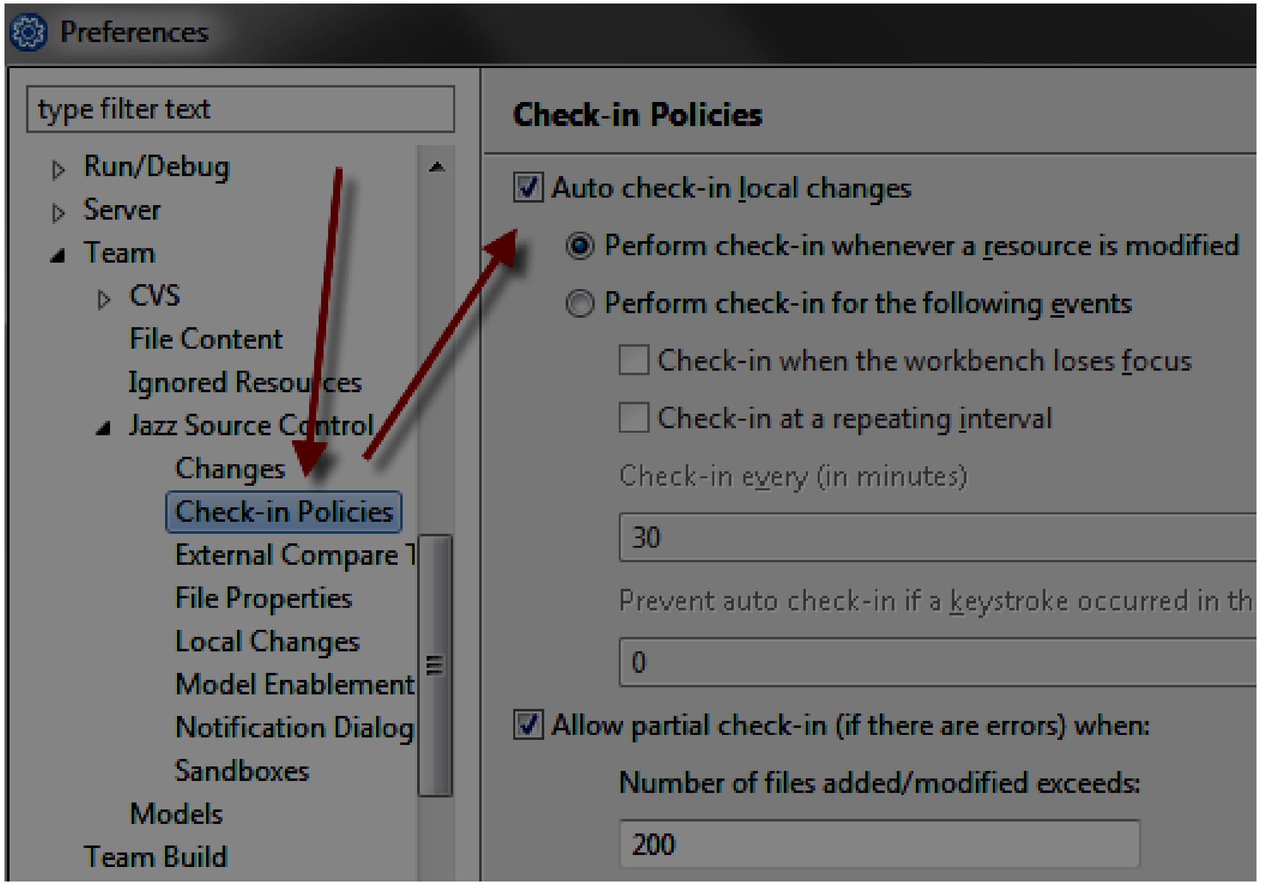
Defining a component and adding files

However, this is painfully slow....

There is a better way....

Auto check-in

- From the RTC "Check-in Policies" preference page, you can select "Auto check-in local changes". When you create a file or folder it will automatically be checked in and associated with a change set.


Preferences

type filter text

- Run/Debug
- Server
- Team
  - CVS
  - File Content
  - Ignored Resources
  - Jazz Source Control
    - Changes
    - Check-in Policies
    - External Compare T
    - File Properties
    - Local Changes
    - Model Enablement
    - Notification Dialog
    - Sandboxes
  - Models
- Team Build

Check-in Policies

[✓] Auto check-in local changes

[●] Perform check-in whenever a resource is modified

[○] Perform check-in for the following events

[ ] Check-in when the workbench loses focus

[ ] Check-in at a repeating interval

Check-in every (in minutes): 30

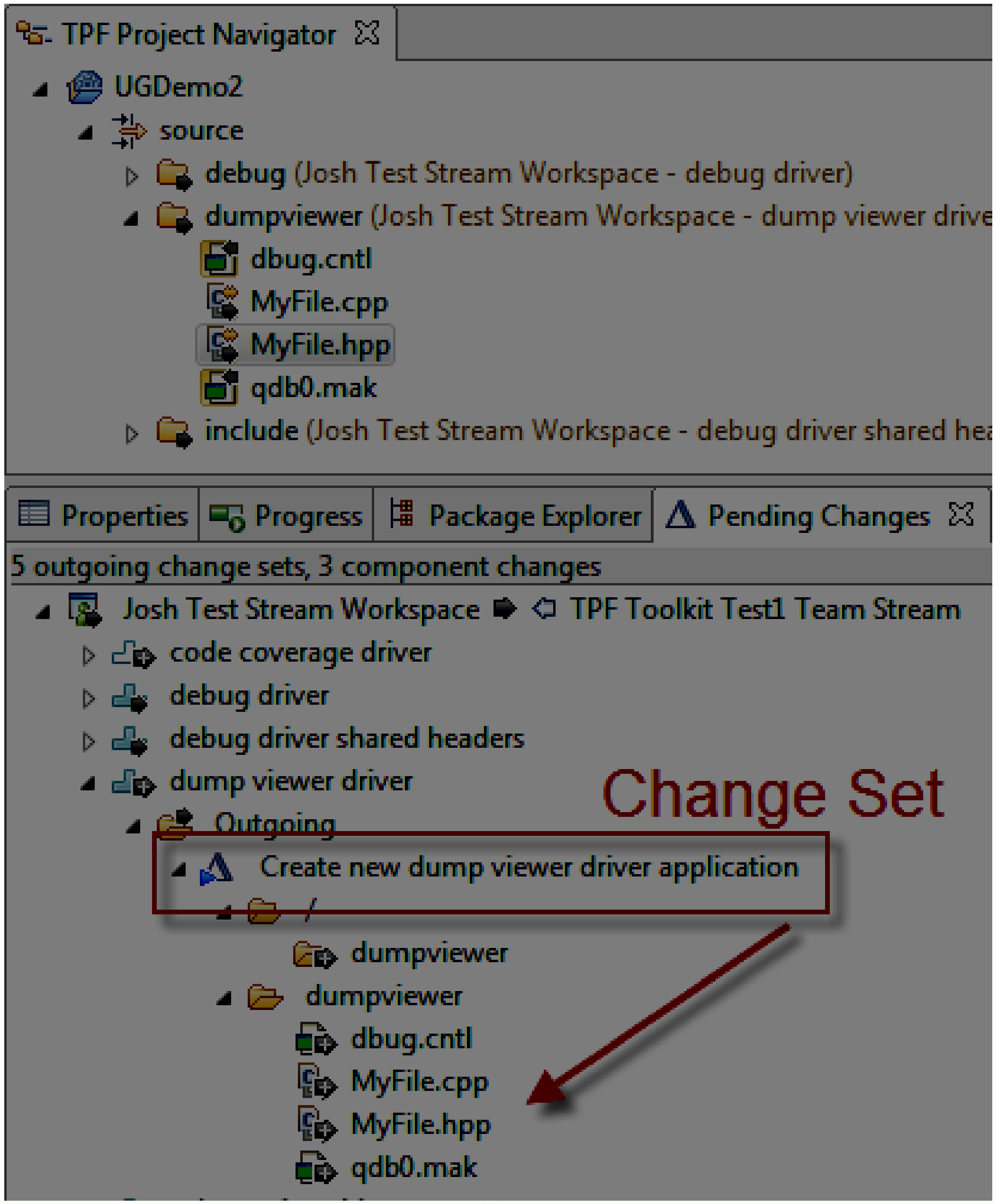
Prevent auto check-in if a keystroke occurred in th: 0

[✓] Allow partial check-in (if there are errors) when:

Number of files added/modified exceeds: 200

Auto check-in

- This simplifies working in the Pending Changes view, eliminates the manual check in step, and ensures the code is backed up on the RTC server. It automatically checked in my unresolved changes.



TPF Project Navigator

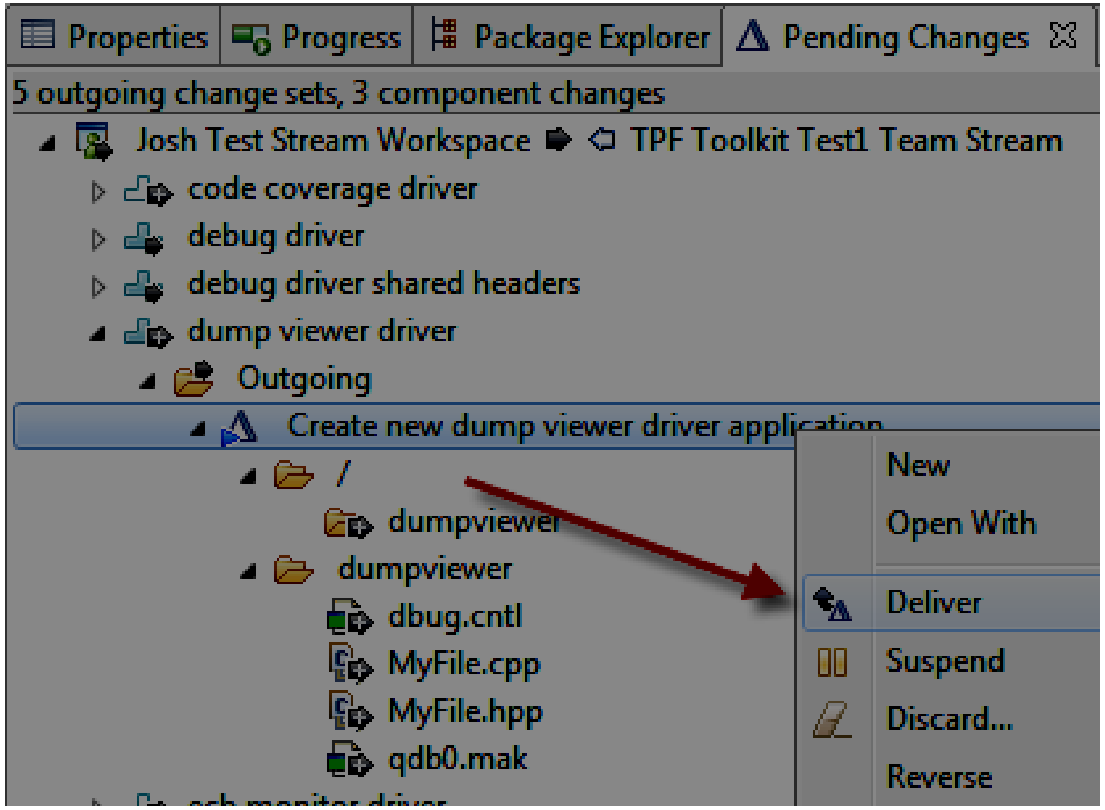
UGDemo2
source
debug (Josh Test Stream Workspace - debug driver)
dumpviewer (Josh Test Stream Workspace - dump viewer drive
dbug.cntl
MyFile.cpp
MyFile.hpp
qdb0.mak
include (Josh Test Stream Workspace - debug driver shared hea

Properties | Progress | Package Explorer | Pending Changes

5 outgoing change sets, 3 component changes
Josh Test Stream Workspace ➡ TPF Toolkit Test1 Team Stream
code coverage driver
debug driver
debug driver shared headers
dump viewer driver
Outgoing
Create new dump viewer driver application
/
dumpviewer
dumpviewer
dbug.cntl
MyFile.cpp
MyFile.hpp
qdb0.mak

Auto check-in

- Note that the "deliver" action commits "change sets" (promotes the source code) to the stream.



5 outgoing change sets, 3 component changes

- Josh Test Stream Workspace ➡ TPF Toolkit Test1 Team Stream
  - code coverage driver
  - debug driver
  - debug driver shared headers
  - dump viewer driver
    - Outgoing
      - Create new dump viewer driver applic
        - /
          - dumpviewer
        - dumpviewer
          - dbug.cntl
          - MyFile.cpp
          - MyFile.hpp
          - qdb0.mak
  - ech monitor driver

Suggested Build set up

- Auto check-in is great, but if you don’t set up your TPF Toolkit projects well, it can have some undesirable consequences. For example, if your “remote working directory” and “remote sandbox” point to the same directory, after running a build, RTC may automatically check in your generated objects, listings and etc. As such, the following set up is suggested for your directories for TPF Toolkit projects:

Suggested Build set up

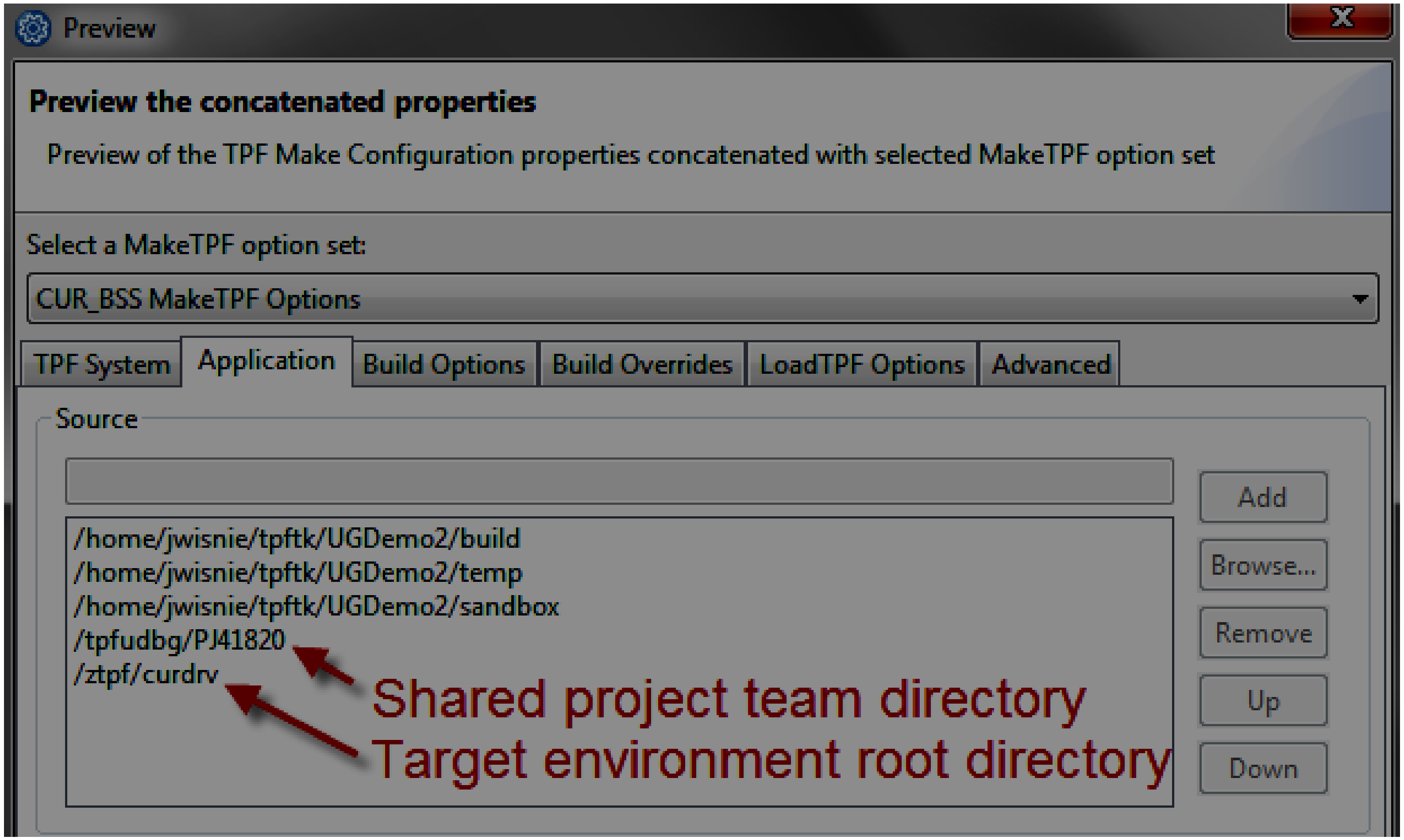
- Create a directory using your project name. This directory will contain all of your work for this project. (ie /home/jwisnie/tpftk/PJXXXXX)
- Point the “remote sandbox” to a sandbox subdirectory in your project directory. The “remote sandbox” is where the source code you create and automatically check-in will be located. (ie /home/jwisnie/tpftk/PJXXXXX/sandbox)
- Point the “remote working directory” to a build subdirectory in your project directory. The “remote working directory” is where your build setup, build results, and etc will be located. (ie /home/jwisnie/tpftk/PJXXXXX/build)

Suggested Build set up

- If desired, create a “temp” directory in your project directory. The “temp” directory is where scaffolding, test and other code will reside. This code will not be checked in or preserved but needs to be part of the build environment. (ie /home/jwisnie/tpftk/PJXXXXX/temp)
- Set up your build path as follows for the optimal experience using auto check in and etc:
  - /home/jwisnie/tpftk/PJXXXXX/build
  - /home/jwisnie/tpftk/PJXXXXX/temp
  - /home/jwisnie/tpftk/PJXXXXX/sandbox
  - Shared project team directories
  - Target environment root directories

Suggested Build set up

- TPF Make Configuration Application Build path example.



Preview the concatenated properties

Preview of the TPF Make Configuration properties concatenated with selected MakeTPF option set

Select a MakeTPF option set: CUR_BSS MakeTPF Options

TPF System | Application | Build Options | Build Overrides | LoadTPF Options | Advanced

Source

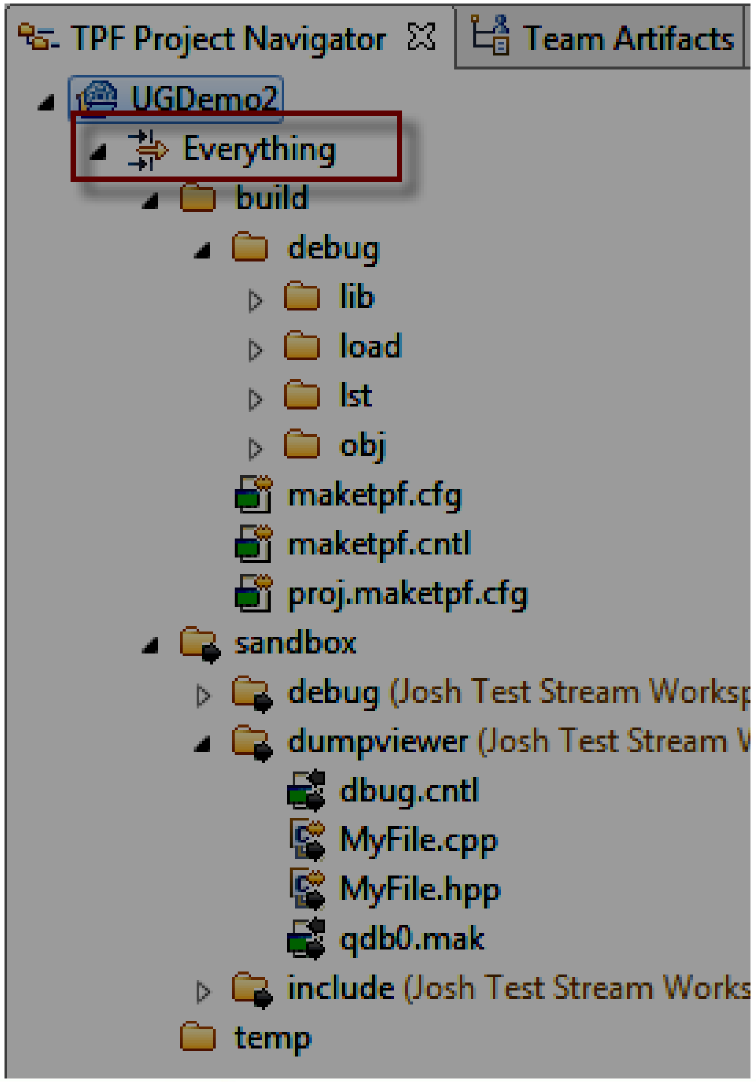
```
/home/jwisnie/tpftk/UGDemo2/build
/home/jwisnie/tpftk/UGDemo2/temp
/home/jwisnie/tpftk/UGDemo2/sandbox
/tpfudbg/PJ41820
/ztpf/curdrv
```

Shared project team directory

Target environment root directory

Suggested Build set up

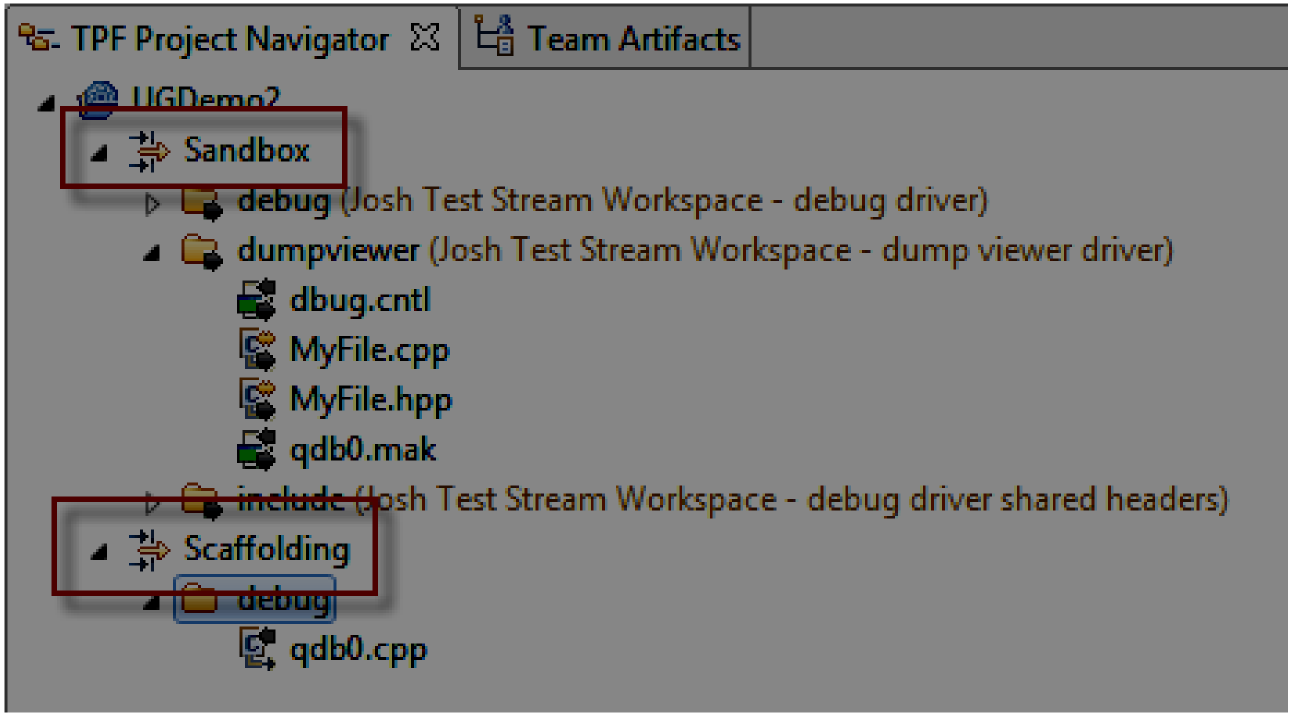
- The TPF Project Navigator view setup depends upon the filters you define. This example filter shows all files from the project root directory.


TPF Project Navigator Team Artifacts

```
UGDemo2
  Everything
    build
      debug
        lib
        load
        lst
        obj
      maketpf.cfg
      maketpf.cntl
      proj.maketpf.cfg
    sandbox
      debug (Josh Test Stream Worksp
      dumpviewer (Josh Test Stream W
        dbug.cntl
        MyFile.cpp
        MyFile.hpp
        qdb0.mak
      include (Josh Test Stream Works
    temp
```

Suggested Build set up

- In this example, two filters show the source code in the sandbox and scaffolding directories.



TPF Project Navigator | Team Artifacts

```
UGDemo2
  Sandbox
    debug (Josh Test Stream Workspace - debug driver)
    dumpviewer (Josh Test Stream Workspace - dump viewer driver)
      dbug.cntl
      MyFile.cpp
      MyFile.hpp
      qdb0.mak
    include (Josh Test Stream Workspace - debug driver shared headers)
  Scaffolding
    debug
      qdb0.cpp
```

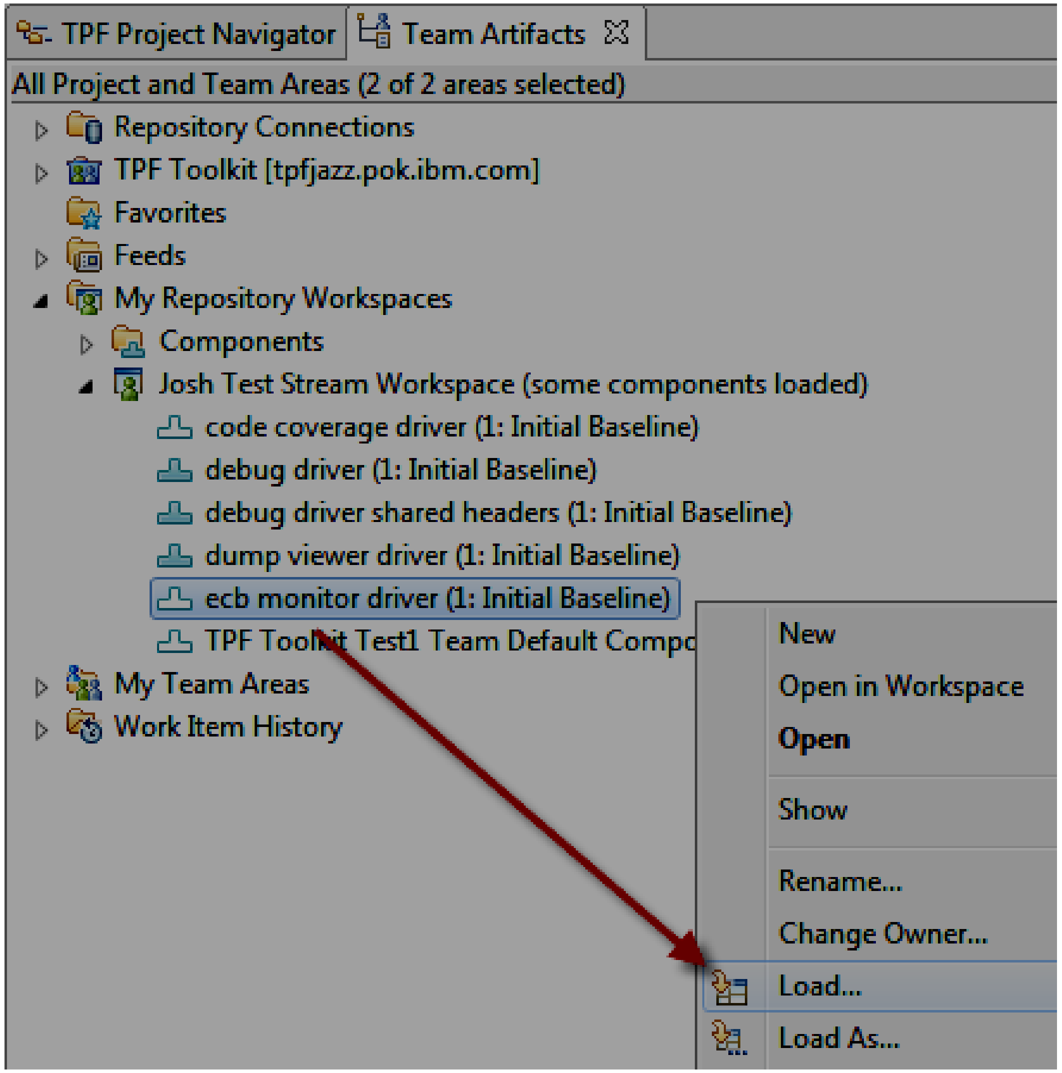
Suggested Build set up

- Given the intricacies of this set up, it may be advisable to create a menu manager or other action to create and set up the project, directory structure, build environment and etc. for your users.

Suggested Build set up: End user example

Assuming your administrator does not automate the process, what does your end user need to do?

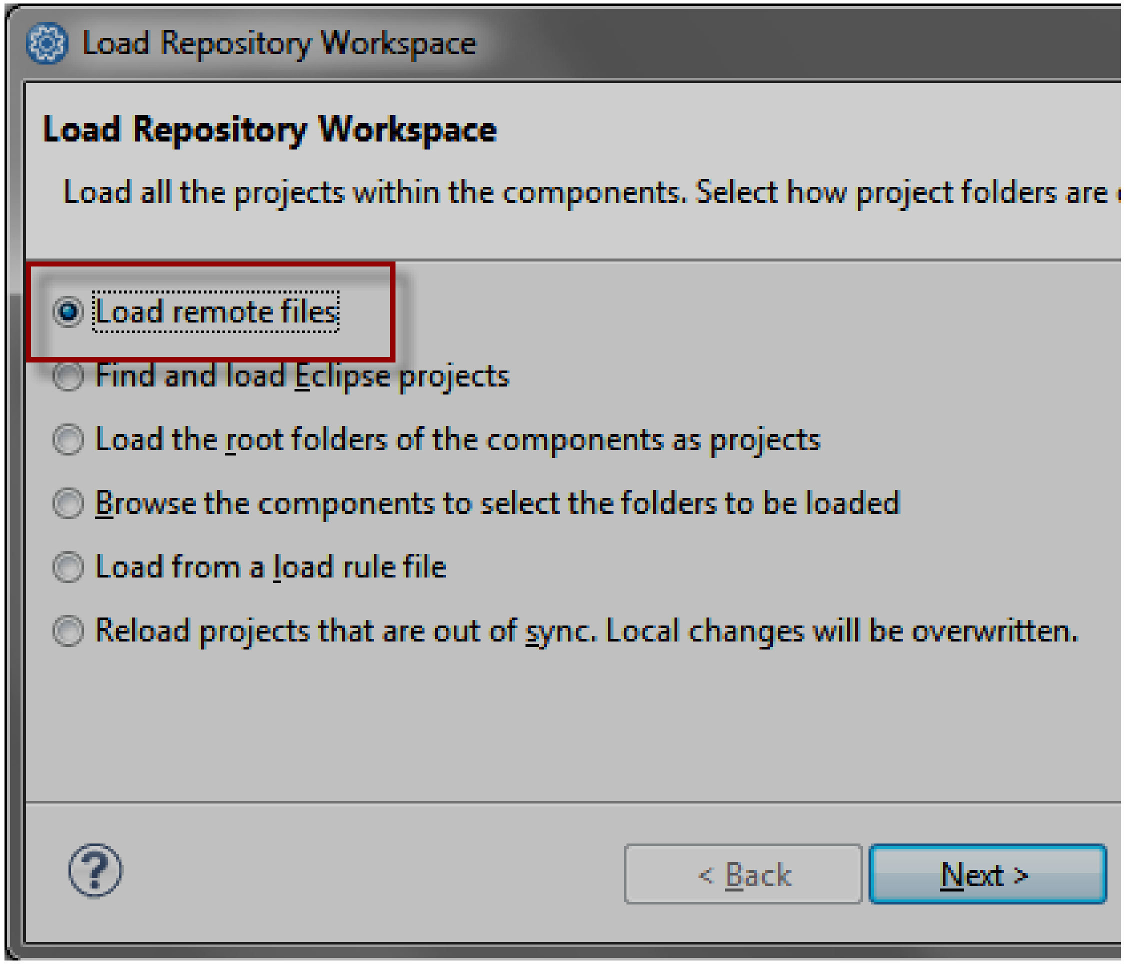
In the Team Artifacts view, load the component containing the code you need to work with.



TPF Project Navigator
Team Artifacts
All Project and Team Areas (2 of 2 areas selected)
Repository Connections
TPF Toolkit [tpfjazz.pok.ibm.com]
Favorites
Feeds
My Repository Workspaces
Components
Josh Test Stream Workspace (some components loaded)
code coverage driver (1: Initial Baseline)
debug driver (1: Initial Baseline)
debug driver shared headers (1: Initial Baseline)
dump viewer driver (1: Initial Baseline)
ecb monitor driver (1: Initial Baseline)
TPF Toolkit Test1 Team Default Compo
My Team Areas
Work Item History
New
Open in Workspace
Open
Show
Rename...
Change Owner...
Load...
Load As...

Suggested Build set up: End user example

In the Load Wizard, select “Load remote files”.



Load Repository Workspace

Load Repository Workspace

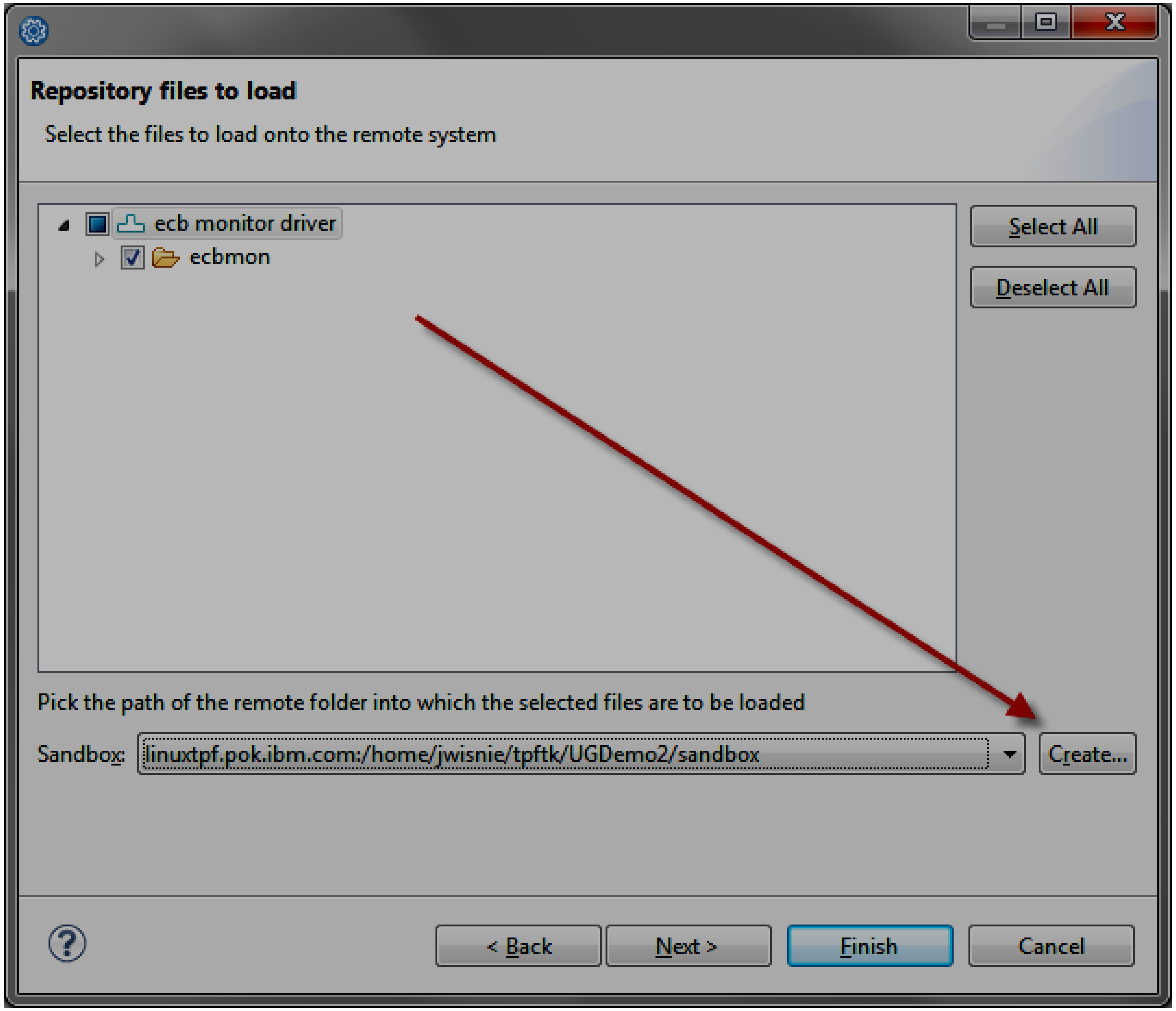
Load all the projects within the components. Select how project folders are

- [✓ Load remote files]
- Find and load Eclipse projects
- Load the root folders of the components as projects
- Browse the components to select the folders to be loaded
- Load from a load rule file
- Reload projects that are out of sync. Local changes will be overwritten.

< Back | Next >

Suggested Build set up: End user example

In the Load Wizard, click the create sandbox button.



Repository files to load

Select the files to load onto the remote system

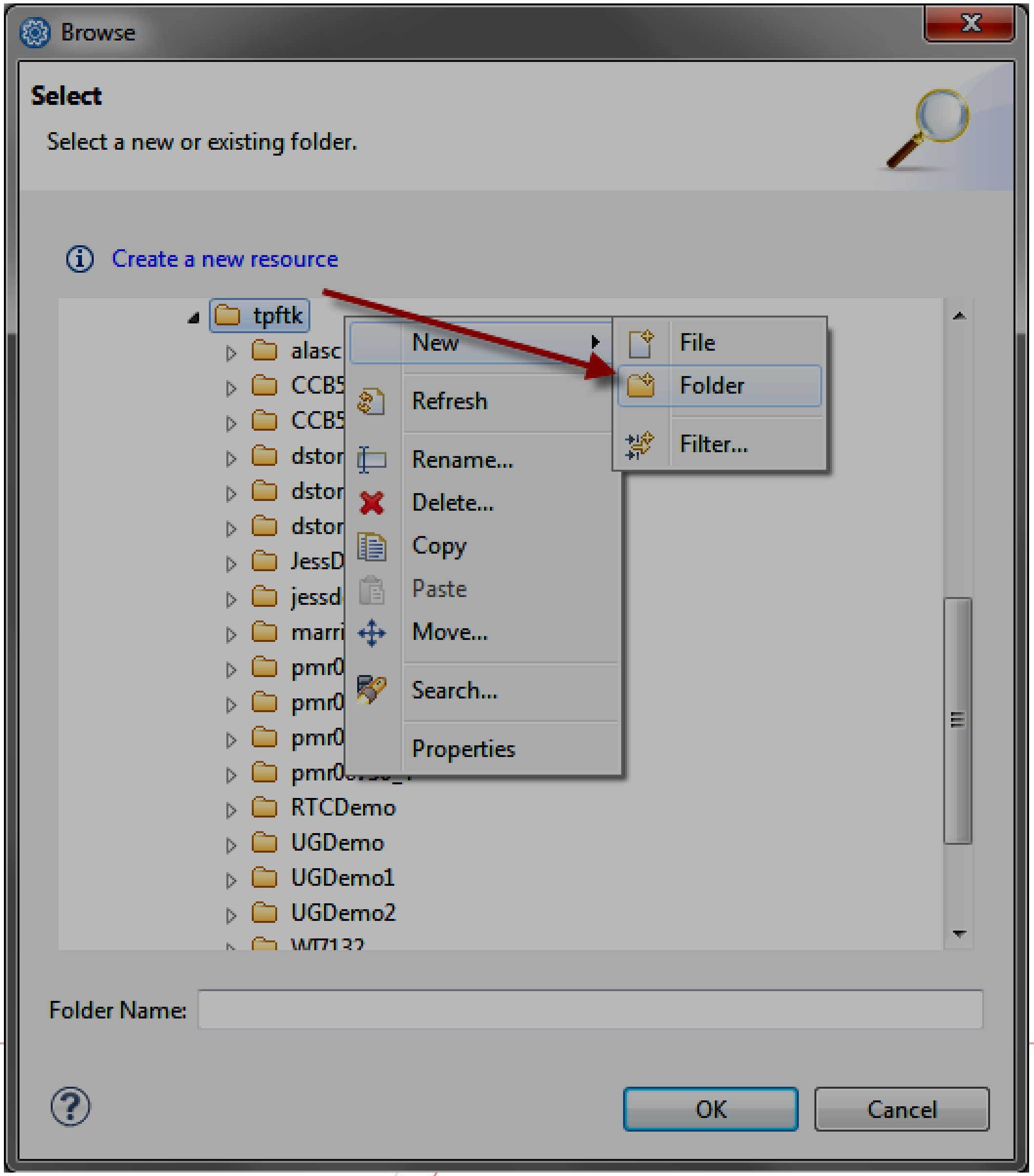
- ecb monitor driver
  - ecbmon

Pick the path of the remote folder into which the selected files are to be loaded

Sandbox: linuxtpf.pok.ibm.com:/home/jwisnie/tpftk/UGDemo2/sandbox

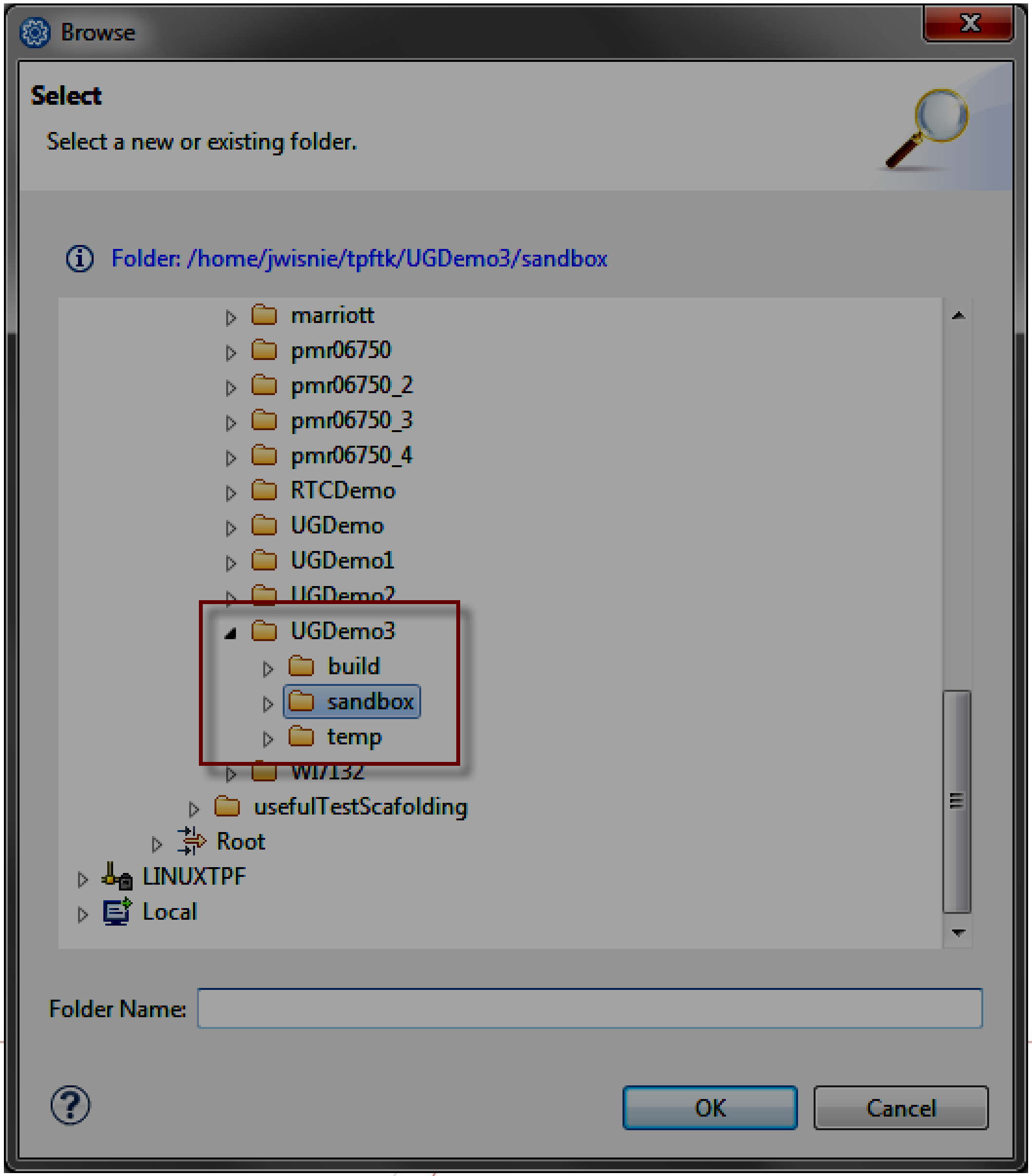
Suggested Build set up: End user example

In the Browse wizard, create the project folder.



Suggested Build set up: End user example

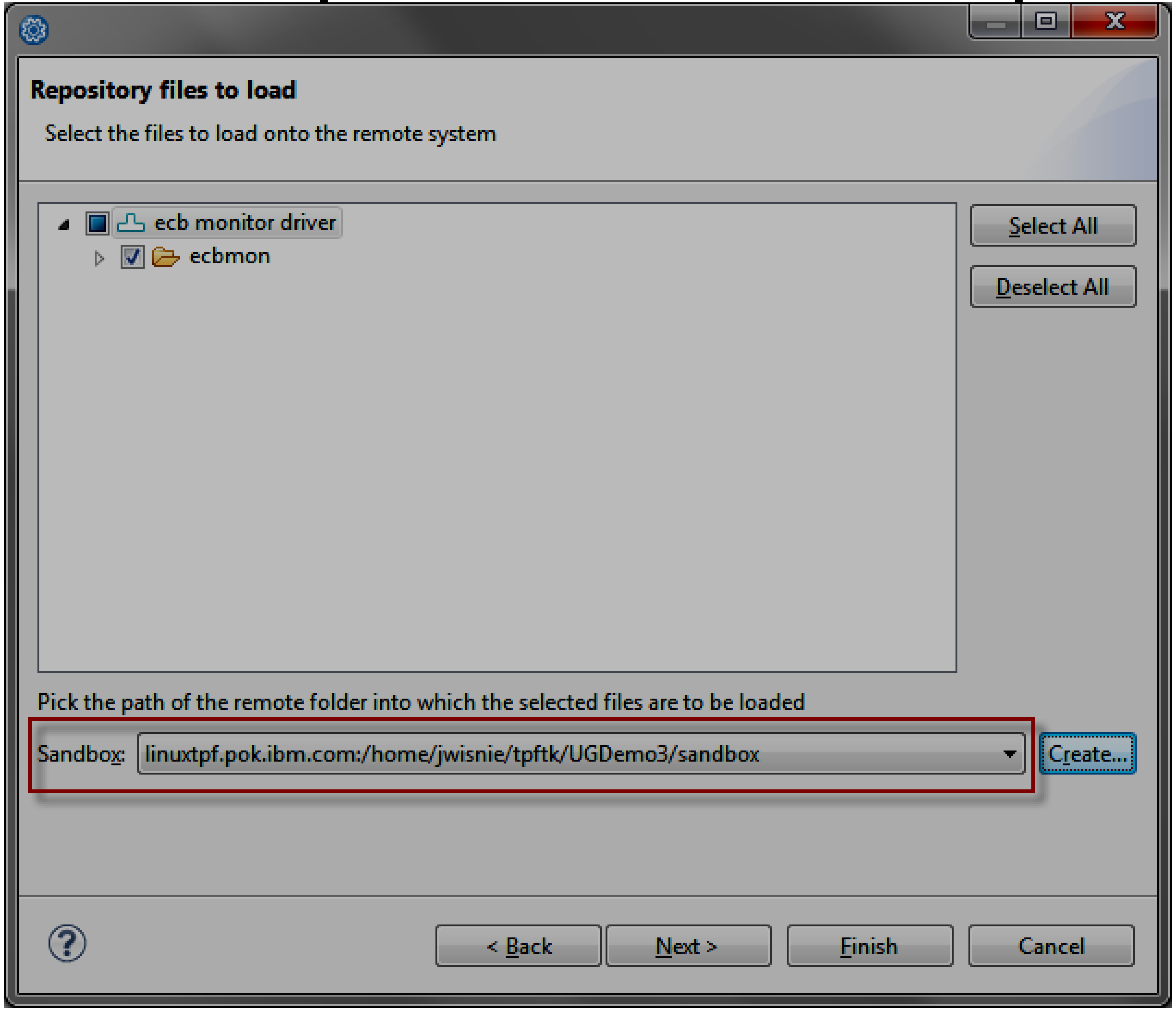
Likewise, create the build, scaffolding and sandbox directories. Select the sandbox subdirectory and choose OK.



Folder: /home/jwisnie/tpftk/UGDemo3/sandbox

Suggested Build set up: End user example

The Load Wizard shows the selected sandbox directory. Click next.


Repository files to load

Select the files to load onto the remote system

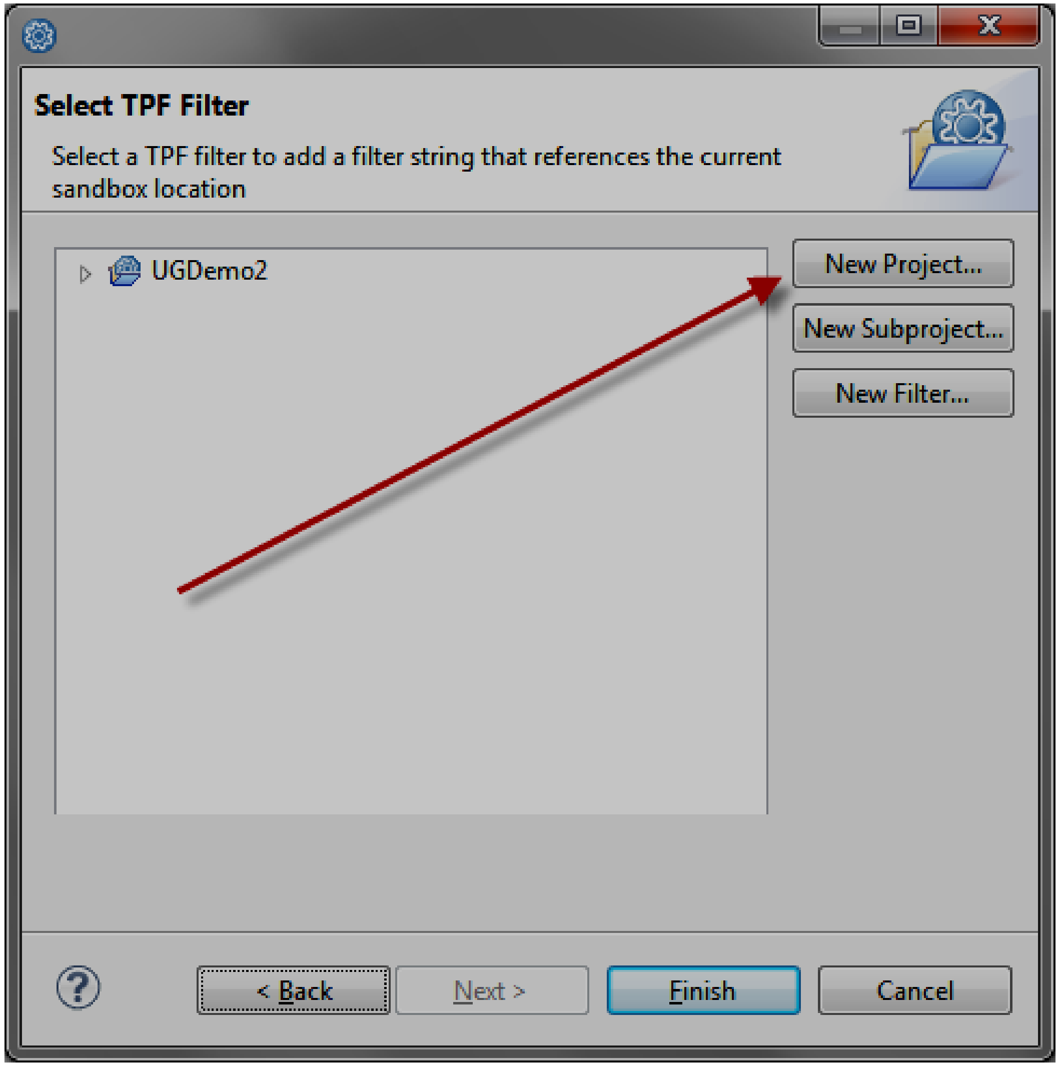
- ecb monitor driver
  - ecbmon

Pick the path of the remote folder into which the selected files are to be loaded

Sandbox: linuxtpf.pok.ibm.com:/home/jwisnie/tpftk/UGDemo3/sandbox

Suggested Build set up: End user example

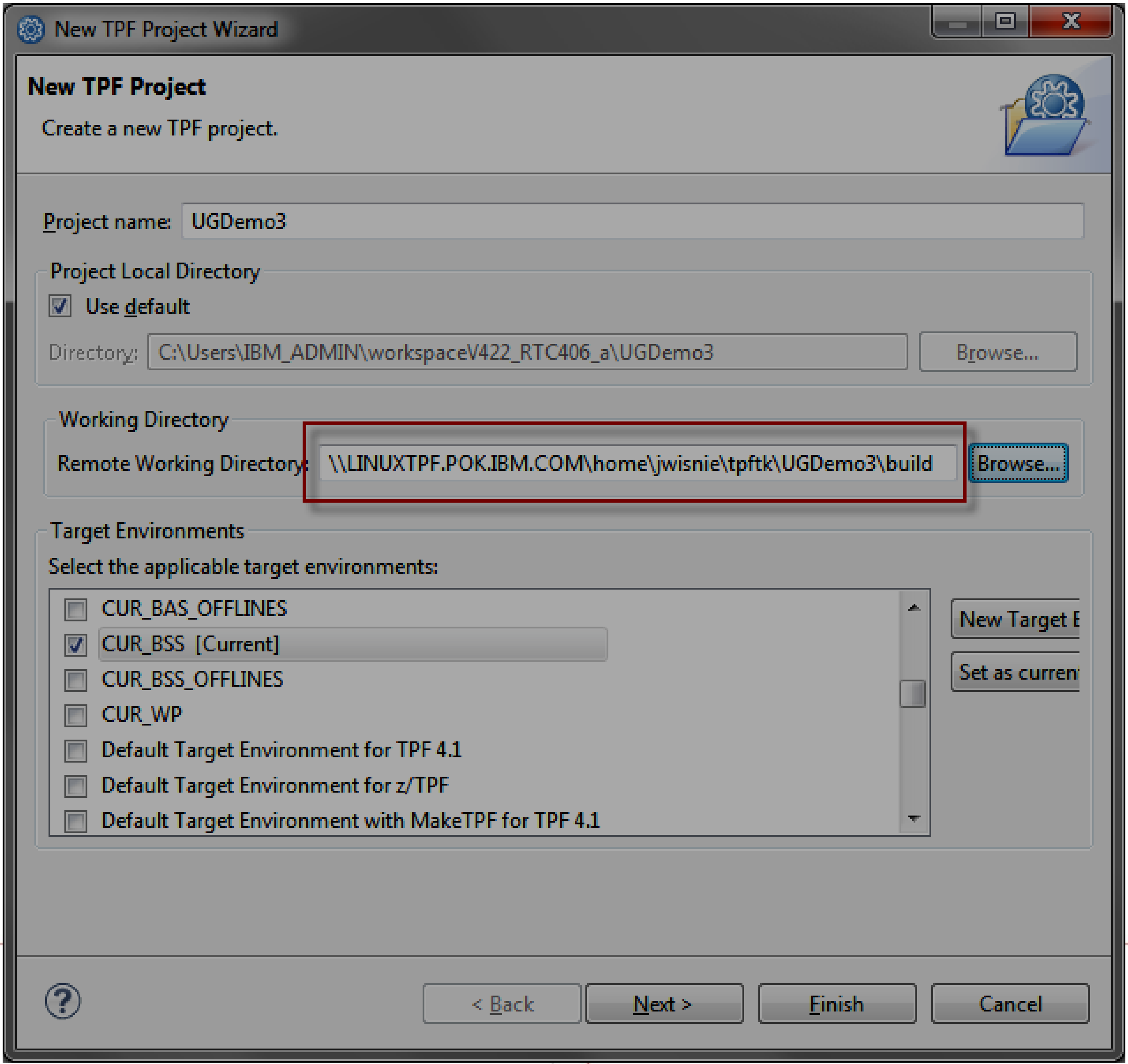
Click the New Project button.


Select TPF Filter

Select a TPF filter to add a filter string that references the current sandbox location

Suggested Build set up: End user example

Set the Remote Working Directory to the new build folder we created earlier. Fill in the other required fields. Click next.



New TPF Project Wizard

New TPF Project

Create a new TPF project.

Project name: UGDemo3

Project Local Directory

[✓] Use default

Directory: C:\Users\IBM_ADMIN\workspaceV422_RTC406_a\UGDemo3

Working Directory

Remote Working Directory: \\LINUXTPF.POK.IBM.COM\home\jwisnie\tpftk\UGDemo3\build

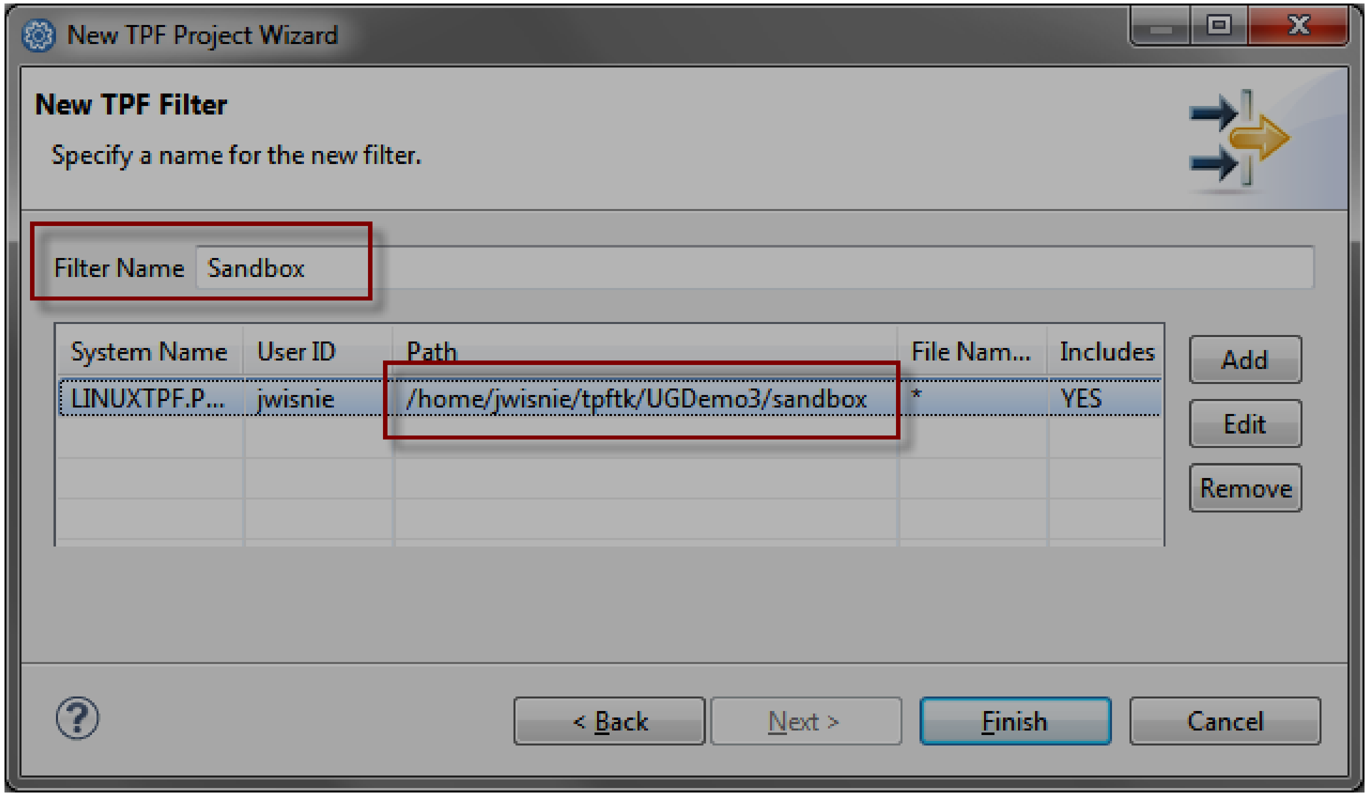
Target Environments

Select the applicable target environments:

- [ ] CUR_BAS_OFFLINES
- [✓] CUR_BSS [Current]
- [ ] CUR_BSS_OFFLINES
- [ ] CUR_WP
- [ ] Default Target Environment for TPF 4.1
- [ ] Default Target Environment for z/TPF
- [ ] Default Target Environment with MakeTPF for TPF 4.1

Suggested Build set up: End user example

Name the filter and point it to the new sandbox folder or the project directory. Click next.



New TPF Filter

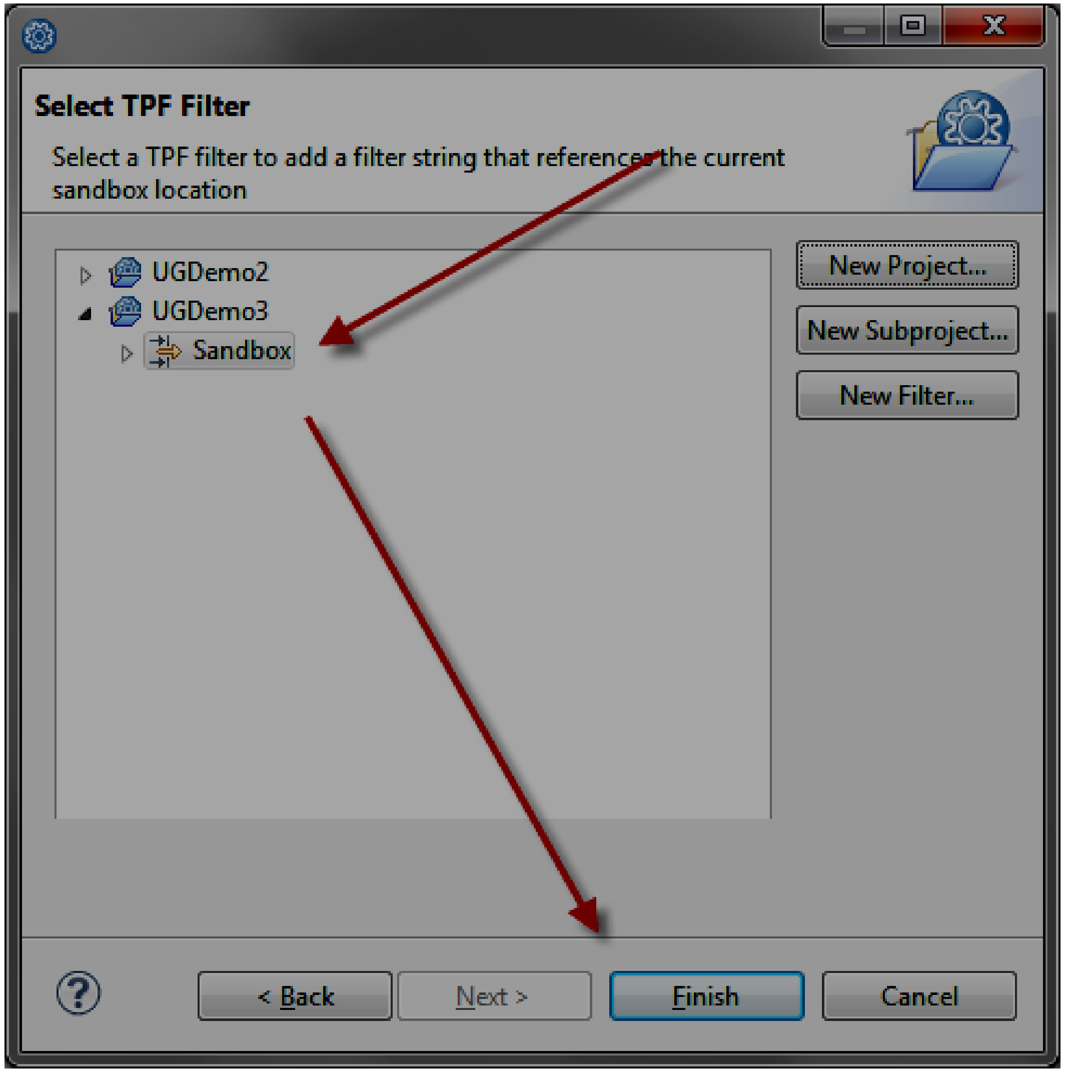
Specify a name for the new filter.

Filter Name: Sandbox

System Name	User ID	Path	File Nam...	Includes
LINUXTPF.P...	jwisnie	/home/jwisnie/tpftk/UGDemo3/sandbox	*	YES

Suggested Build set up: End user example

Select the new filter under the new TPF Project and click Finish.

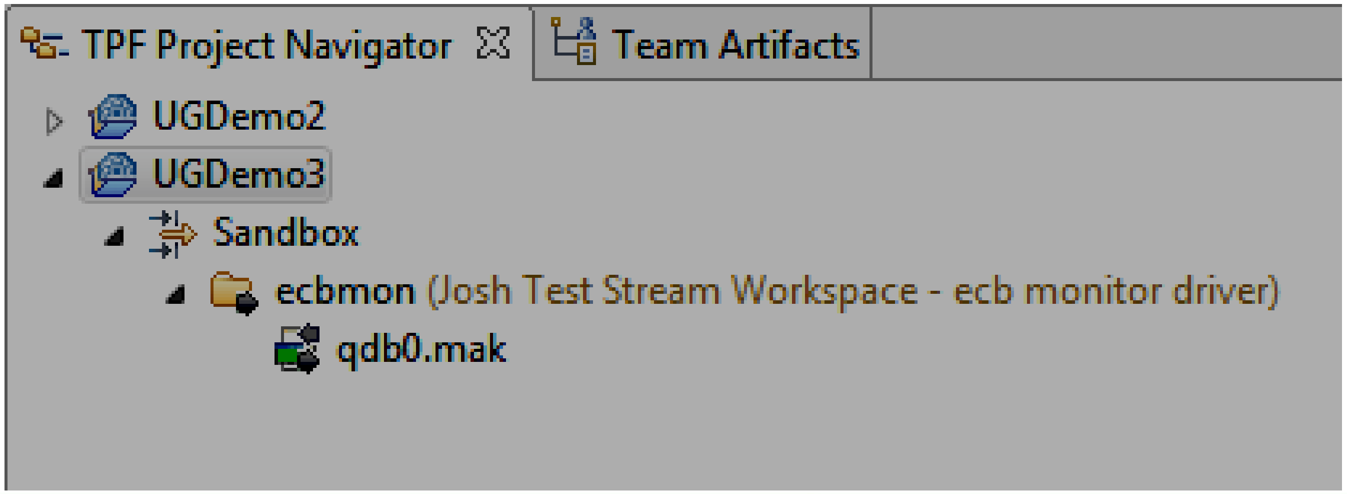


Select TPF Filter

Select a TPF filter to add a filter string that references the current sandbox location

Suggested Build set up: End user example

The TPF project is now created. The component is loaded and ready for the user to make code changes. Set up your TPF Make Configuration Application Build path as previously described. Build, load and test as normal. Use the Pending Changes view to deliver your code.

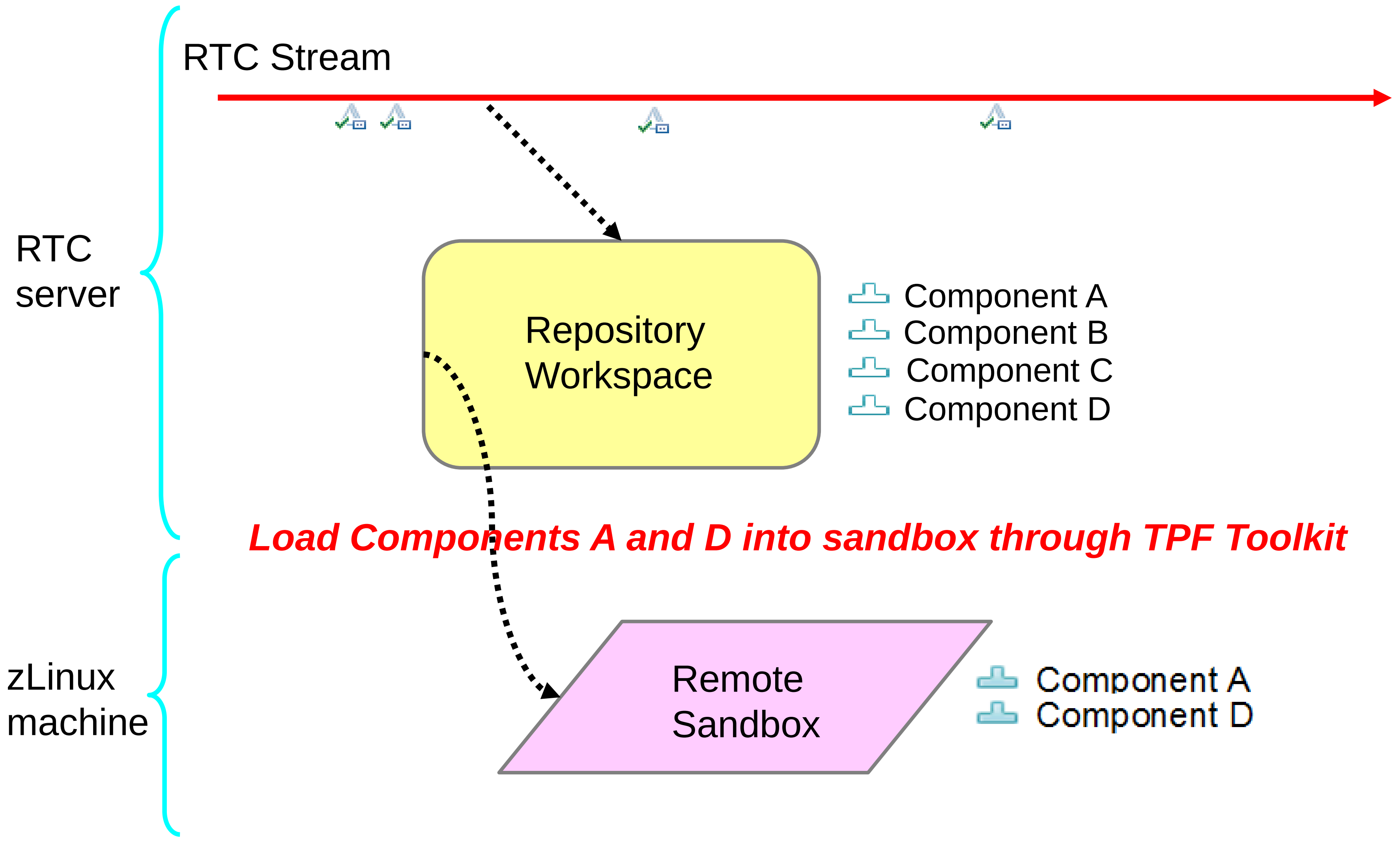


Questions?

Trademarks

Notes

Files per component for performance



Load Components A and D into sandbox through TPF Toolkit