

z/TPF EE V1.1

z/TPFDF V1.1

TPF Toolkit for WebSphere® Studio V3

TPF Operations Server V1.2



IBM Software Group

## *TPF Users Group Spring 2006*

### Single Source Conversion using IBM TPF Toolkit

Andrea Rice

Applications Migration Best Practices – Part 2

**AIM Enterprise Platform Software**

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

© IBM Corporation 2006

Any references 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.

# Single Source

- Goal: Create and maintain one set of source code that will run on TPF 4.1 and on z/TPF until migration is complete.
- Single Source APARs introduce changes into TPF 4.1 that are required for z/TPF
  - Allows the changes to be made to applications on TPF 4.1
  - Don't need two copies of application source.
  - Enables the same application source to be built for TPF 4.1 or z/TPF with minimal conditional code.
  - See migration guide for a list of single source APARs
- Other Single Source Changes
  - Not all required / recommended changes have an associated APAR.

# Rules

- Rules are used to identify individual single source changes
- Each rule is identified with a unique Rule ID
- Rule ID conventions
  - For changes associated with an APAR, the rule ID is:  
APAR + 1 character rule identifier  
Example: PJ29957a
  - For “Other” changes that are not associated with an APAR, the rule ID is:  
OTR + 4 character change description + 1 character rule identifier  
Example: OTRPACKb

# Rule Attributes

- Detectable / Not Detectable
- Definite / Potential
- Error / Warning
- Auto Fix / Manual Fix

# Detectable / Not Detectable

- Detectable – Something the TPF Toolkit can detect by scanning C/CPP or HLASM source code.

Example:

PJ29576a – Flags variables declared as decimal data type. For example,

```
decimal(15,5) op_1 = 245680.98786d
```

- Not Detectable – Something that requires manually examining code.

Example: No rule to detect applications that store floating point numbers on file as hexadecimal data.

# Definite / Potential

- Definite – Something that is definitely a single source problem that must be addressed.

Example: PJ29593a – TPF Header files moved to a tpf/ subdirectory. For example,

```
#include <tpfapi.h>
```

When this rule is automatically fixed, the result is:

```
#include <tpf/tpfapi.h>
```

- Potential – Something that might be a single source error.

Example: PJ29436a – Detects programs using ebw012. However manual examination is required to determine if the program was activated as a result of an AOR call.

# Error / Warning

- Error – Something that if it is not fixed will break code on z/TPF.

Example: PJ29957b – The LC\_TOD category is not supported on z/TPF. For example:

```
string = setlocale(LC_TOD, valid_locale_name);
```

- Warning – Something that if not fixed will cause a compiler warning on z/TPF.

Example: OTRPRECa - Use parentheses to dictate order in expressions. For example:

```
Warning:          if (a || b && c)
No Warning:      if (a || (b && c))
```

# Auto Fix / Manual Fix

- Auto Fix – The TPF Toolkit can automatically repair the single source error.

Example: OTRLONGa – This rule can automatically replace long data types with int. For example:

```
long myvar = 5;    =>    int myvar = 5;
```

- Manual Fix – Manual code changes required to resolve the single source error.

Example: PJ29957a – If you are using the LC\_TOD category, you must manually update source code to use the TZ environment variable instead.

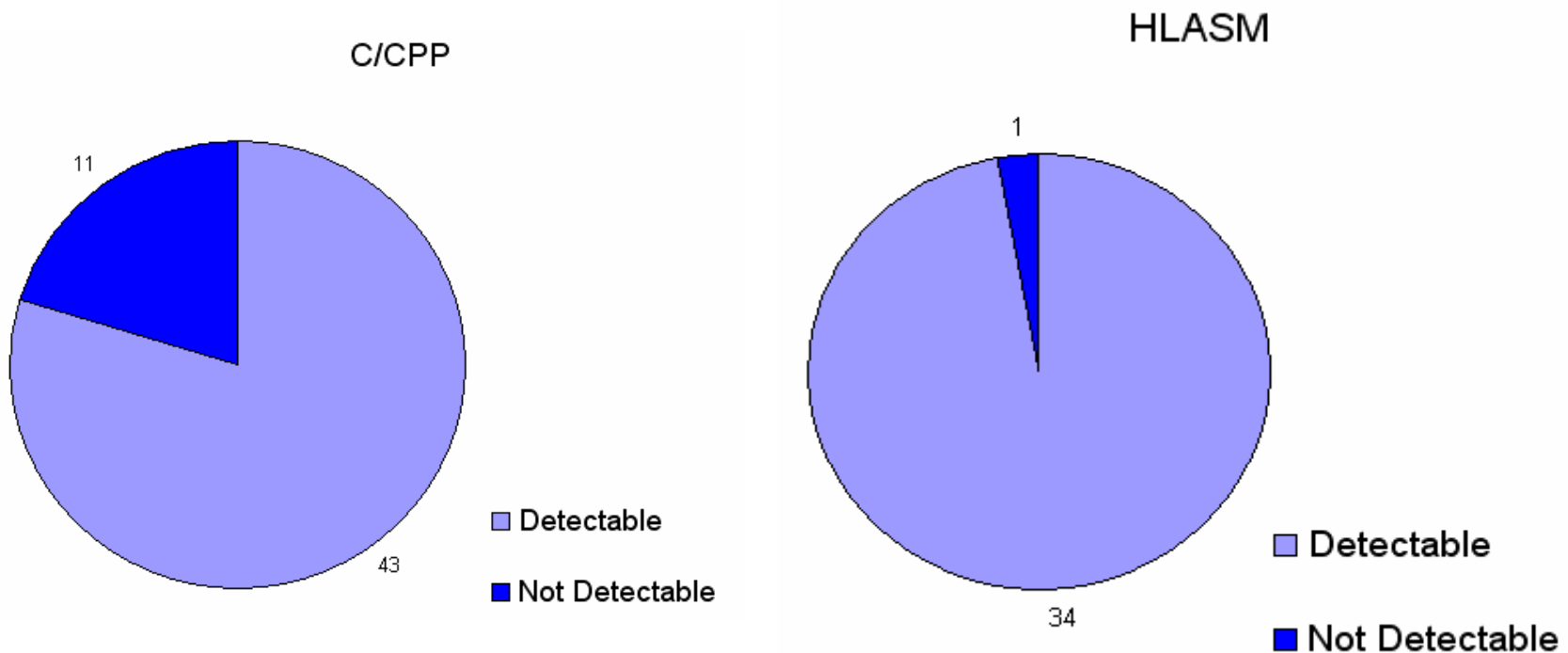


# Statistics & Rules

- Rules will not be matched evenly throughout code
  - Some code may have no errors, some code may have lots.
  - Some files may have a lot of errors that can be fixed automatically, other code might have lots of errors that need to be fixed manually.
  - Rule distributions amongst various characteristics will not represent error distributions in source code.

# Statistics

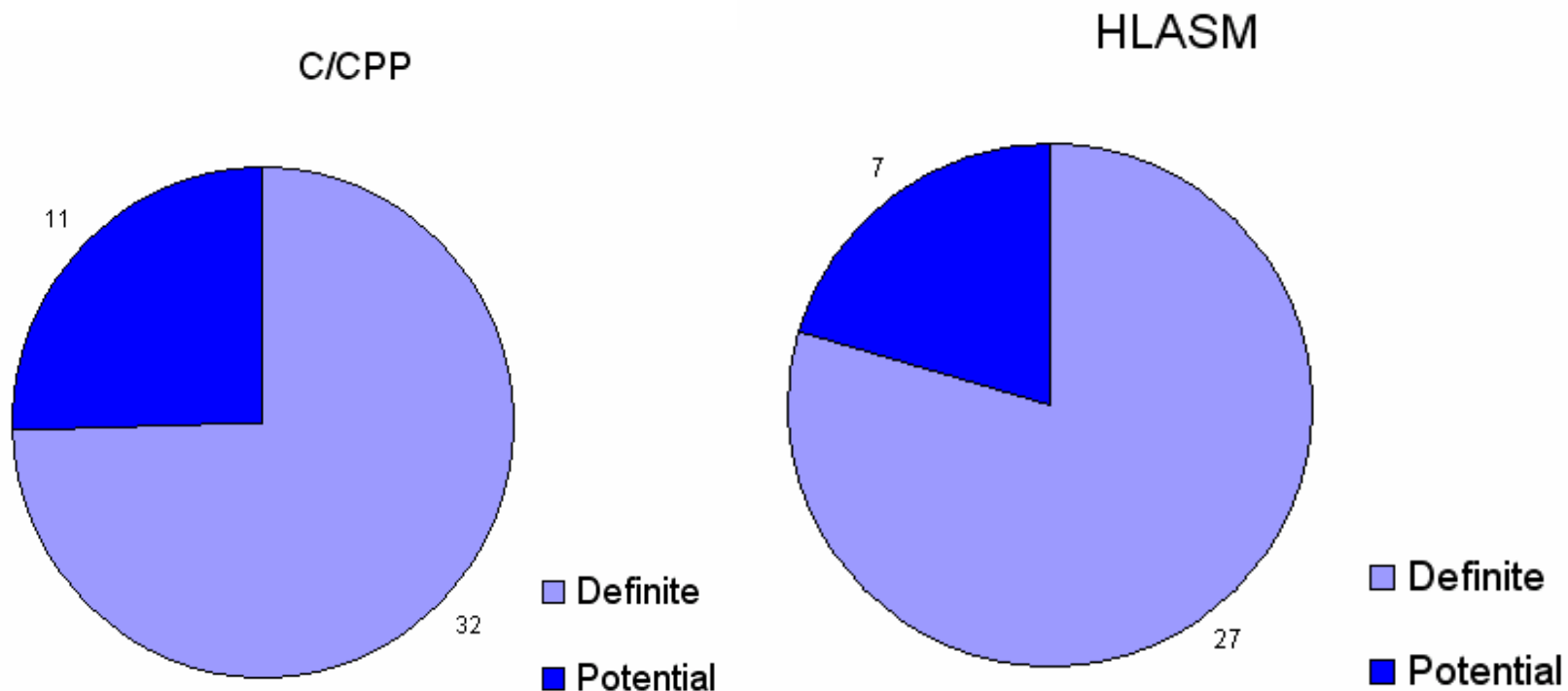
- Detectable / Not Detectable
  - Number of rules vs. Number of issue types that require manual detection.



Note: Data based on Interim Fix v3.0.6 Rule Set

# Statistics

- Potential / Definite
  - Number of rules that detect definite problems vs. Number of rules that detect potential errors.

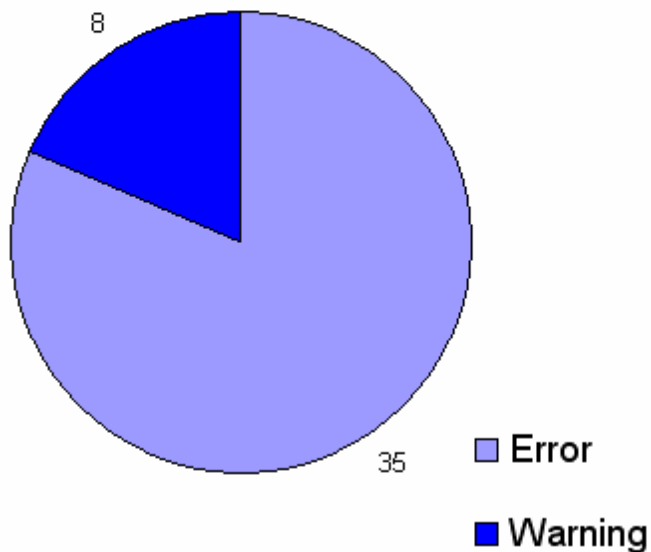


Note: Data based on Interim Fix v3.0.6 Rule Set

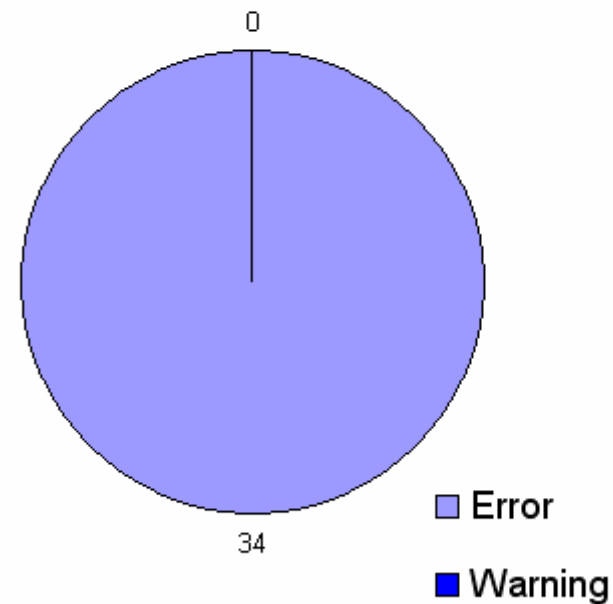
# Statistics

- Errors / Warnings
  - Number of rules that flag warnings vs. number of rules that flag errors.

C/CPP



HLASM

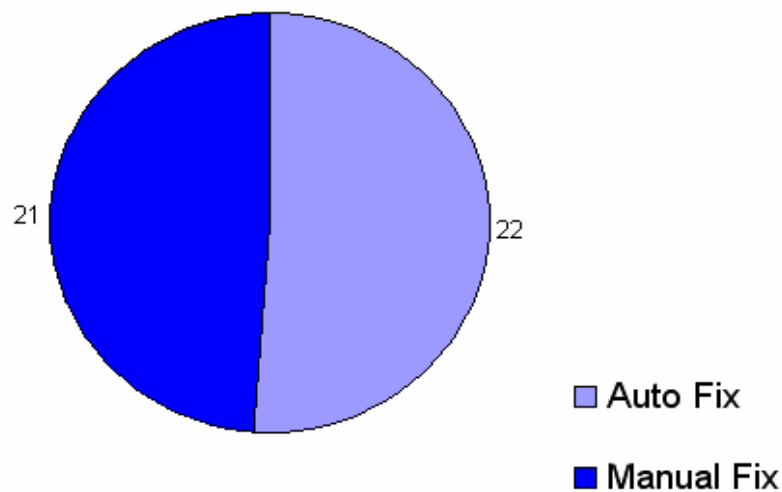


Note: Data based on Interim Fix v3.0.6 Rule Set

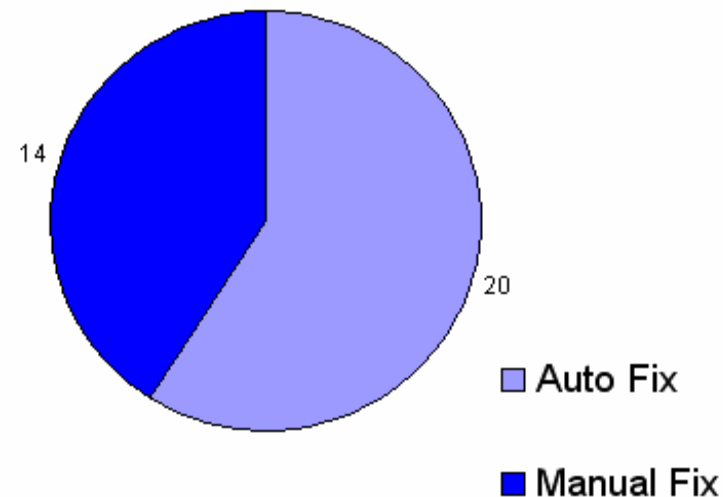
# Statistics

- Auto Fix / Manual Fix
  - Number of rules that flag problems that can be fixed automatically vs. Number of rules that flag problems that require manual fixes.

C/CPP



HLASM



Note: Data based on Interim Fix v3.0.6 Rule Set

# More Information

- TPF Toolkit Help
  - Section: Installing, migrating, and configuring > Migrating > Migrating from TPF 4.1 to z/TPF > Converting to single source
  - Page: Rules for scanning
    - Table 1. Single source APARs and associated rules
    - Table 2. Single source rules not associated with an APAR
    - Table 3. Undetected changes

# More Information

- Individual Rule Help Pages

## **PJ29436a - Use ebw024 to access address of data in programs activated by AOR calls**

This rule is associated with APAR PJ29436.

For detailed information on this single source APAR, see [APAR PJ29436](#).

### **Purpose**

When an application is activated as a result of one of the following activate\_on\_receipt (AOR) socket APIs, the address of data is saved:

- activate\_on\_receipt
- activate\_on\_receipt\_with\_length
- activate\_on\_receipt\_of\_TCP\_message
- activate\_on\_receipt\_of\_TCP\_message2

On TPF 4.1, a 4 byte address is required and the data is saved in EBW012 - EBW015. On z/TPF, an 8 byte address is required and the data is saved starting at ebw024.

### **Detection**

This rule looks for **ebw012**.

For example:

```
(void)memcpy(&read_addr, &(ecbptr()->ebw012), sizeof(read_addr));
```

...

# Rule Selection

- Preference Page: TPF Toolkit > Migration to z/TPF > Rules

Rule Activation

| Enable                              | Rule ID  | Description   | Language | Type |
|-------------------------------------|----------|---|----------|------|
| <input checked="" type="checkbox"/> | PJ29436a | Change ebw012 to ebw024 in programs activa...       | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29575a | Flag all pointers that might be used in assemble... | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29576a | gcc compiler does not support decimal data typ...   | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29576b | gcc compiler does not support decimal data typ...   | C\CPP    |      |
| <input type="checkbox"/>            | PJ29576c | gcc compiler does not support decimal data typ...   | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29576d | Flag include statements for decimal.h               | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593a | Use tpf header files in tpf subdirectory.           | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593b | Remove \$ symbol from all header file names.        | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593c | Update #include statements to use _ instead o...    | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593d | Use angle brackets <> in include statements         | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593e | cs() (compare-and-swap) function is no longer l...  | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29593f | cds() (compare-double-and-swap) function is n...    | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29630a | Types time_t, size_t and ssize_t changed from ...   | C\CPP    |      |
| <input type="checkbox"/>            | PJ29630b | Function calls passing types time_t, size_t, and... | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29937a | Some system header files moved to sys subdir...     | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29957a | setlocale function changed for obsolete catego...   | C\CPP    |      |
| <input checked="" type="checkbox"/> | PJ29957b | Replace usage of locale category LC_TOD with...     | C\CPP    |      |
| <input type="checkbox"/>            | PJ29974- | Function calls for lcs() and lcs10() changed to...  | C\CPP    |      |



# Command Line Invocation

- TFFtool commands
  - `tpftool -s MigrationScan`
  - `tpftool -s MigrationAction`
- Quick & Easy way to find and fix problems
  - [Demo](#)

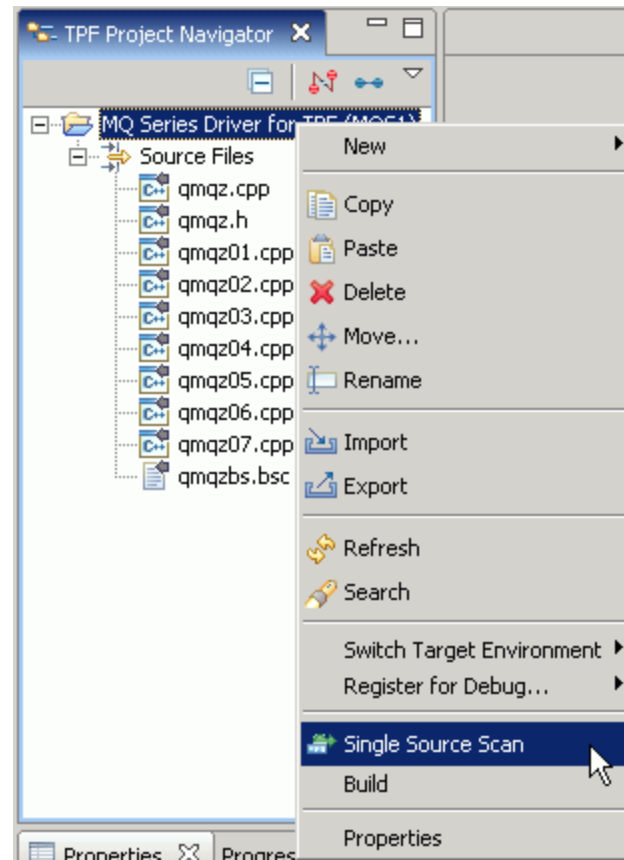
# Leveraging TPF Toolkit with GUI

- Command line tools are easy to use but don't offer any review features
- General Flow



# Invoking Scans

- Scan
  - Project
  - Filter
  - Folder
  - File



# Results





- Remote Error List View lists results









Filter matched 99 of 99 messages

| ID       | Message                                      | S... | Line | Location                           |
|----------|--|------|------|------------------------------------|
| PJ29593a | TPF-unique header files moved to /include... | 10   | 22   | /u/vaide/arice/Sample Code/MQS1/sc |
| PJ29937a | sysgtime.h moved and/or renamed to sys/...   | 10   | 29   | /u/vaide/arice/Sample Code/MQS1/sc |
| PJ29593a | TPF-unique header files moved to /include... | 10   | 30   | /u/vaide/arice/Sample Code/MQS1/sc |
| PJ29593a | TPF-unique header files moved to /include... | 10   | 31   | /u/vaide/arice/Sample Code/MQS1/sc |
| PJ29593a | TPF-unique header files moved to /include... | 10   | 33   | /u/vaide/arice/Sample Code/MQS1/sc |
| OTRLONGa | Use int instead of long because long chan... | 10   | 567  | /u/vaide/arice/Sample Code/MQS1/sc |
| PJ29593a | TPF-unique header files moved to /include... | 10   | 26   | /u/vaide/arice/Sample Code/MQS1/sc |
| OTRLONGa | Use int instead of long because long chan... | 10   | 332  | /u/vaide/arice/Sample Code/MQS1/sc |
| OTRLONGa | Use int instead of long because long chan... | 10   | 583  | /u/vaide/arice/Sample Code/MQS1/sc |

# Icons

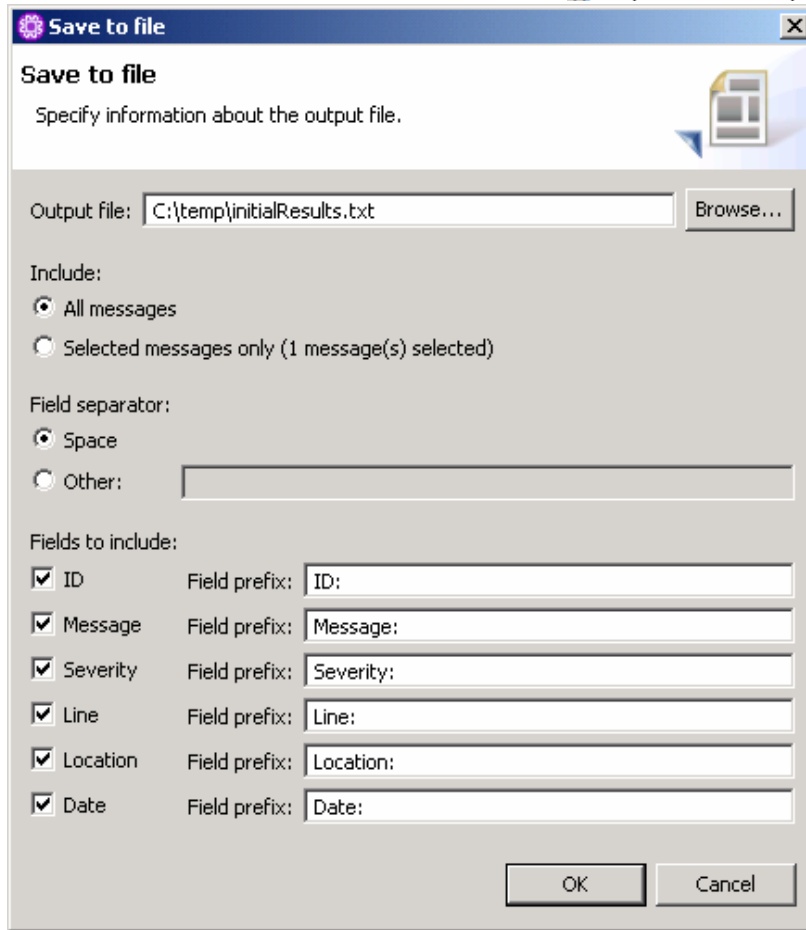
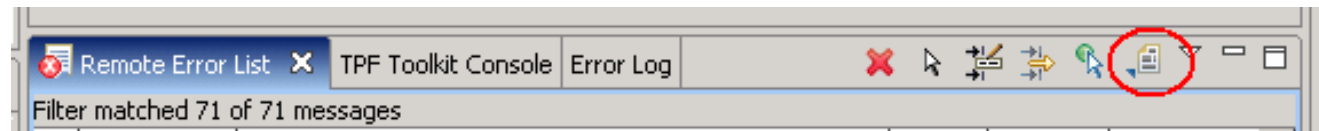
- Icon Legend

-  Fixable
-  Potential
-  Error
-  Warning


|   |                            |
|---|----------------------------|
|    | Error                      |
|    | Error with Fix             |
|    | Potential Error            |
|    | Potential Error with Fix   |
|   | Warning                    |
|  | Warning with Fix           |
|  | Potential Warning          |
|  | Potential Warning with Fix |

# Results

- Saving Results

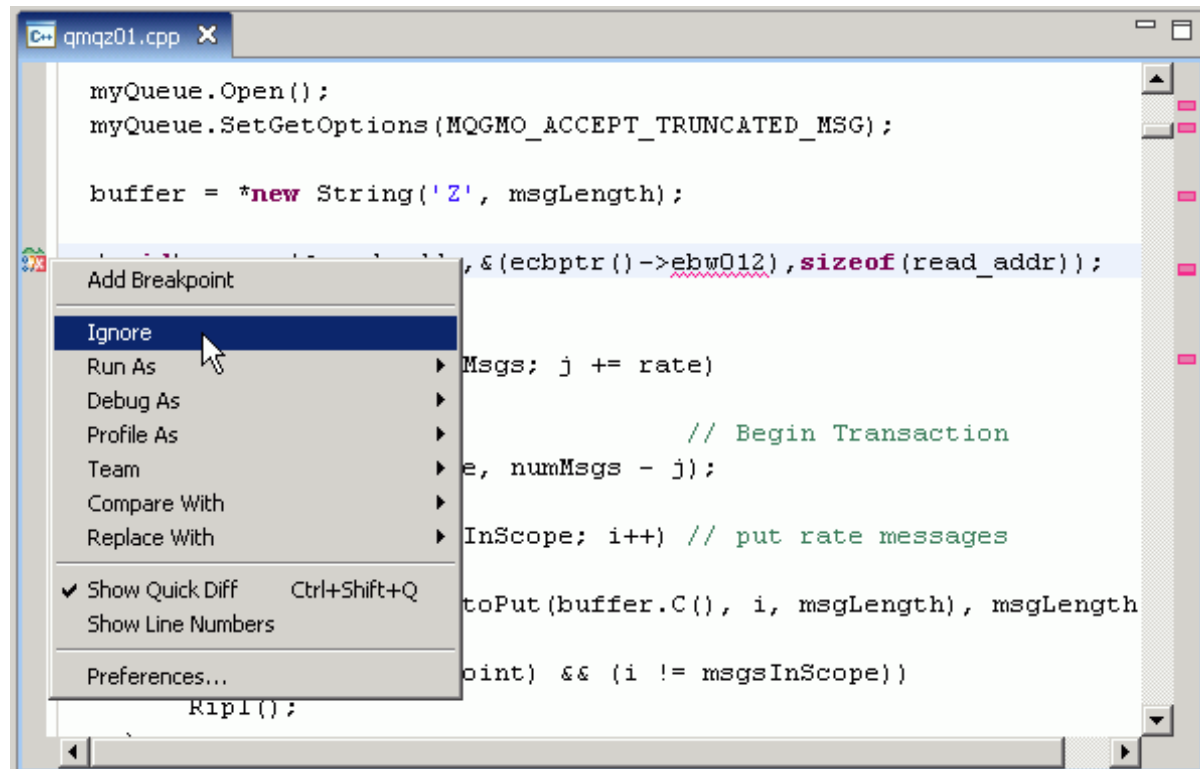


# Quick Fix

- Available while Editing
  - Look for light bulb 
  - [Demo](#)

# Potential Errors

- Open File
  - Quick Fix or Manual Fix
  - Ignore





# Compare Files

- Compare file with migrated version action
  - Allows you to compare your file with a fixed version of the same file.
  - Features Include
    - Comments
      - Preference Page: TPF Toolkit > Migration to z/TPF > Actions
    - Potential Error Handling
      - Preference Page: TPF Toolkit > Migration to z/TPF > Actions
    - White space
      - Tip: General > Compare / Patch preference page > Ignore white space

# Commenting

- Preserving Original Lines
  - Preference Page: TPF Toolkit > Migration to z/TPF > Actions
  - Examples:

On

```
/*#include <cbnode.h>*/  
#include <tpf/cbnode.h>
```

Off

```
#include <tpf/cbnode.h>
```

# Commenting

- Flagging changed lines
  - Re-uses auto comment language profiles (Preference Page: Auto Comment)
    - Tip: Make sure the language profile you use supports a long enough auto comment length.
  - Base Comment set on preference page: TPF Toolkit > Migration to z/TPF > Actions
  - Examples:

On (Base Comment = &RULEID)

```
/*#include <cbnode.h>*/  
#include <tpf/cbnode.h>                                     /*PJ29593a*/
```

Off

```
/*#include <cbnode.h>*/  
#include <tpf/cbnode.h>
```

# Compare Example

The screenshot displays a 'C Compare' window with two panes. The left pane shows the original source code, and the right pane shows the modified source code. The changes are highlighted in blue.

```
/*
/*  PLOG      DATE      DESC
/*  ----      ----      ----
/*  TDxxxxxx  mm/dd/yy
/*****
#include <cmqz.h>
#include <c$mqs.h>

#define DEFAULT_MSG_BUFFER 200
static void Case1(), Case2(),
static void Case5(), Case6()

/*
/*  PLOG      DATE      D
/*  ----      ----      -
/*  TDxxxxxx  mm/dd/yy
/*****
#include <cmqz.h>
/*#include <c$mqs.h> */
#include <tpf/c_mqs.h>

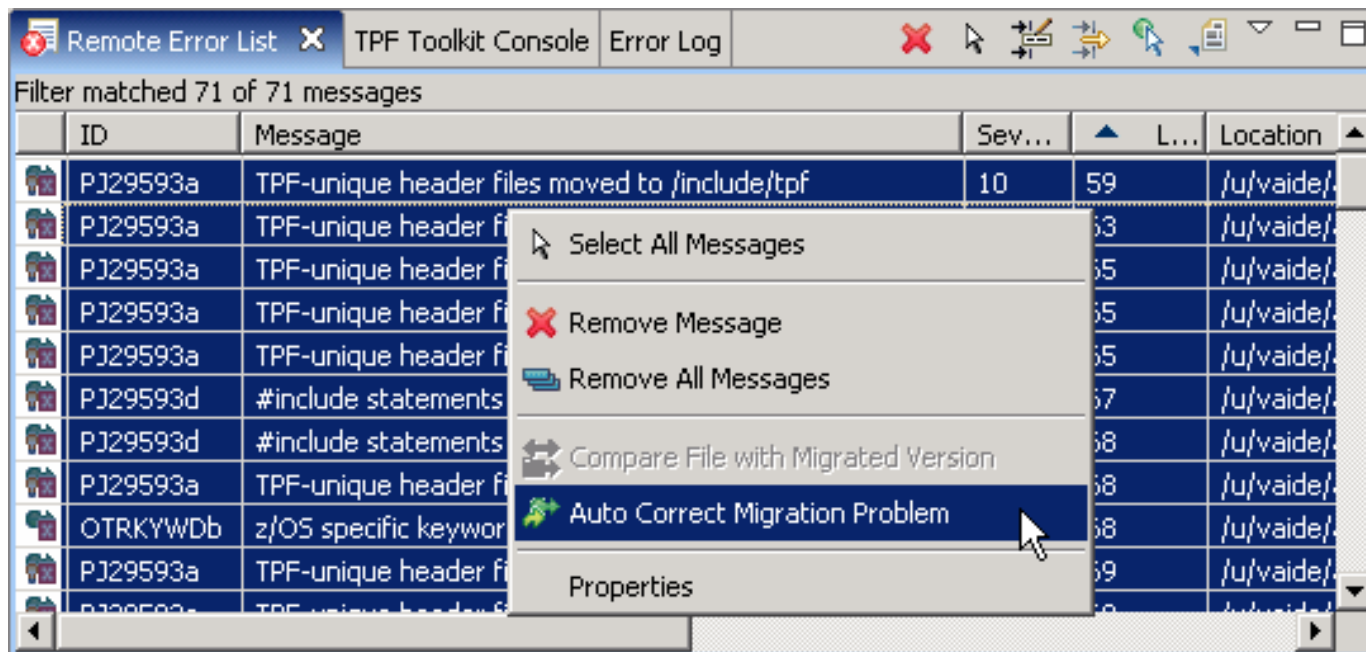
#define DEFAULT_MSG_BUFFER 2
static void Case1(), Case2()
```

# Auto Correct Action

- Automatically updates files with fixes
  - Features Include
    - Commenting (Preference Page: TPF Toolkit > Migration to z/TPF > Actions)
    - Potential Error Handling (Preference Page: TPF Toolkit > Migration to z/TPF > Actions)
    - Restore Files (Preference Page: TPF Toolkit > Migration to z/TPF > Restore Files)

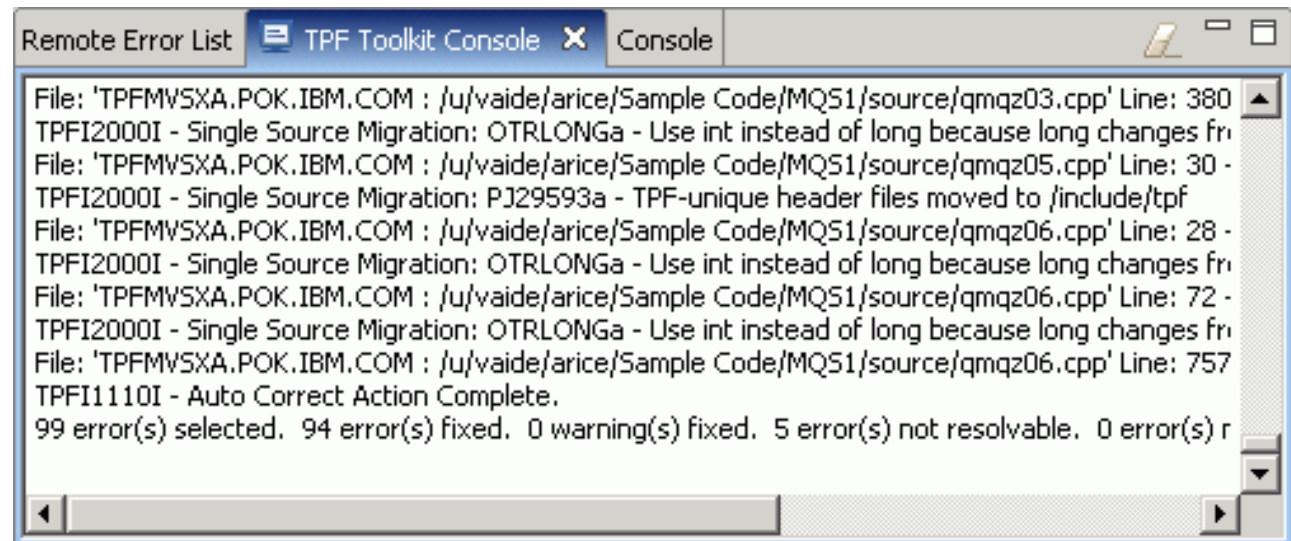
# Auto Correct

- Invoke Example:



# Auto Correct

- Output Example:



```
Remote Error List | TPF Toolkit Console | Console
File: 'TPFMV5XA.POK.IBM.COM : /u/vaide/arice/Sample Code/MQ51/source/qmqz03.cpp' Line: 380
TPFI2000I - Single Source Migration: OTRLONGa - Use int instead of long because long changes fr
File: 'TPFMV5XA.POK.IBM.COM : /u/vaide/arice/Sample Code/MQ51/source/qmqz05.cpp' Line: 30 -
TPFI2000I - Single Source Migration: PJ29593a - TPF-unique header files moved to /include/tpf
File: 'TPFMV5XA.POK.IBM.COM : /u/vaide/arice/Sample Code/MQ51/source/qmqz06.cpp' Line: 28 -
TPFI2000I - Single Source Migration: OTRLONGa - Use int instead of long because long changes fr
File: 'TPFMV5XA.POK.IBM.COM : /u/vaide/arice/Sample Code/MQ51/source/qmqz06.cpp' Line: 72 -
TPFI2000I - Single Source Migration: OTRLONGa - Use int instead of long because long changes fr
File: 'TPFMV5XA.POK.IBM.COM : /u/vaide/arice/Sample Code/MQ51/source/qmqz06.cpp' Line: 757
TPFI1110I - Auto Correct Action Complete.
99 error(s) selected. 94 error(s) fixed. 0 warning(s) fixed. 5 error(s) not resolvable. 0 error(s) r
```

Message:

TPFI1110I - Auto Correct Action Complete.  
99 error(s) selected. 94 error(s) **fixed**. 0 warning(s) fixed. 5  
error(s) **not resolvable**. 0 error(s) not resolved due to **overlap**. 0  
**potential** error(s) ignored.

# Staying Single Source Compatible

- Validation
  - Automatically scans changed files.
  - Settings
    - Preference: TPF Toolkit > Migration to z/TPF
    - Project Properties



# Demo

- Single source conversion using TPF Toolkit
  - Demo

## z/TPF C/C++ Linux Migration Tool

The prototype to the TPF Toolkit updates is a Linux command line migration utility. The Linux command line migration utility is being provided “as-is”.

This set of scripts perform very basic, simple conversion substitutions with TPF 4.1 C/C++ code as input, outputting C/C++ code which should compile under tpf-gcc, versions 3.4 and higher.

### Features:

- Covers about 70% of the C/C++ migration rules
- Runs on the Linux command line
- Runs in a batch mode
- Based on standard Linux utilities like **bash**, **sed**, **awk**
- Can easily be extended to include user defined rules

### Operation:

- Each rule is contained within its own script **cvt\_xyz**
- The master script **cvt\_start** pipes each source file sequentially through all conversion scripts
- A logfile collects all warnings and errors
- The output destination contains the same directory tree as the input directory, with all C/C++ files converted.

### Example:

```
cvt_start /tpf41/apps /ztpf/apps
```

# Single Source Summary

- Available Scan/Fix Methods for C/C++ Applications
  - tpftool command line API for TPF Toolkit
  - TPF Toolkit GUI (Recommended)
  - As-is Linux command line migration utility
- Resources
  - TPF Toolkit Help
  - TPF Info Center – Migration Guide

# Application Migration Summary

1. Move your application source to Linux
2. Change your application source builds to use TPF Make
3. Convert to Single Source
4. Compile code on TPF 4.1
5. Compile code on z/TPF
6. Build & Test on TPF 4.1
7. Build & Test on z/TPF

## Trademarks

IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries

Other company, product, or service names may be trademarks or service marks of others.

### Notes

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.