TPF Toolkit

TPF Users Group - Fall 2009
Migration Tooling in TPF Toolkit

Ankit Pasricha
Main Tent

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

- **Overview**
- **Source Scan**
  - *Analysis* – Planning & Estimation
  - *Implementation*
  - *Validation* – Maintaining Single Source Compliance
- **Resources**
- **Questions**

Overview

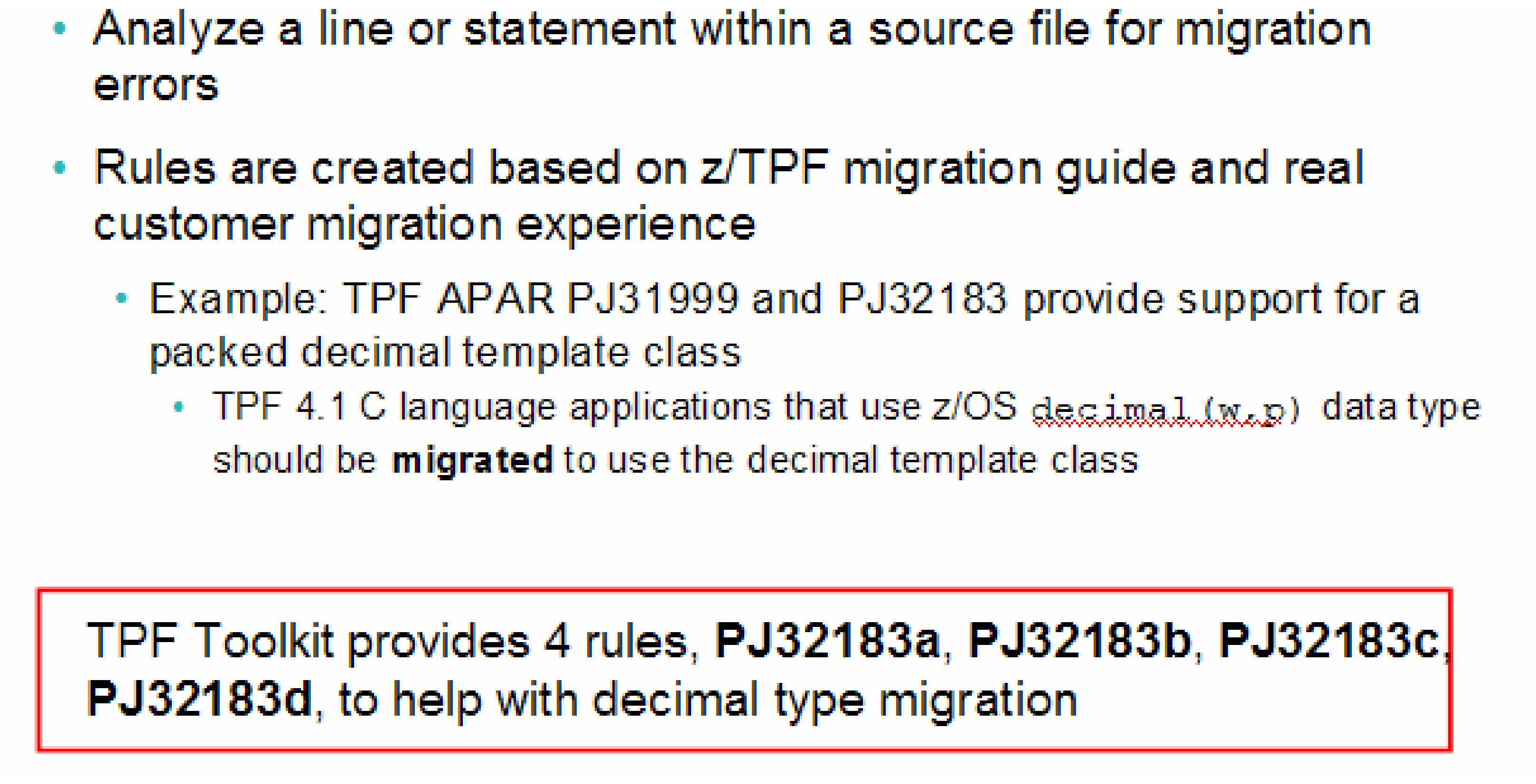
Migration Tooling in TPF Toolkit = **Source Scan**

Overview

- **Source Scan first introduced in TPF Toolkit V3 (2005)**
- **What does Source Scan provide?**
  - Scan your source code for migration problems using **rules** determined by IBM.
  - **Single source**: Fix detected problems so that your source code can be compiled on TPF 4.1 and z/TPF
  - **Extensibility:** Add your own rules to TPF Toolkit to detect and fix migration problems specific to your environment

Overview: Rules

- **What is a rule?**

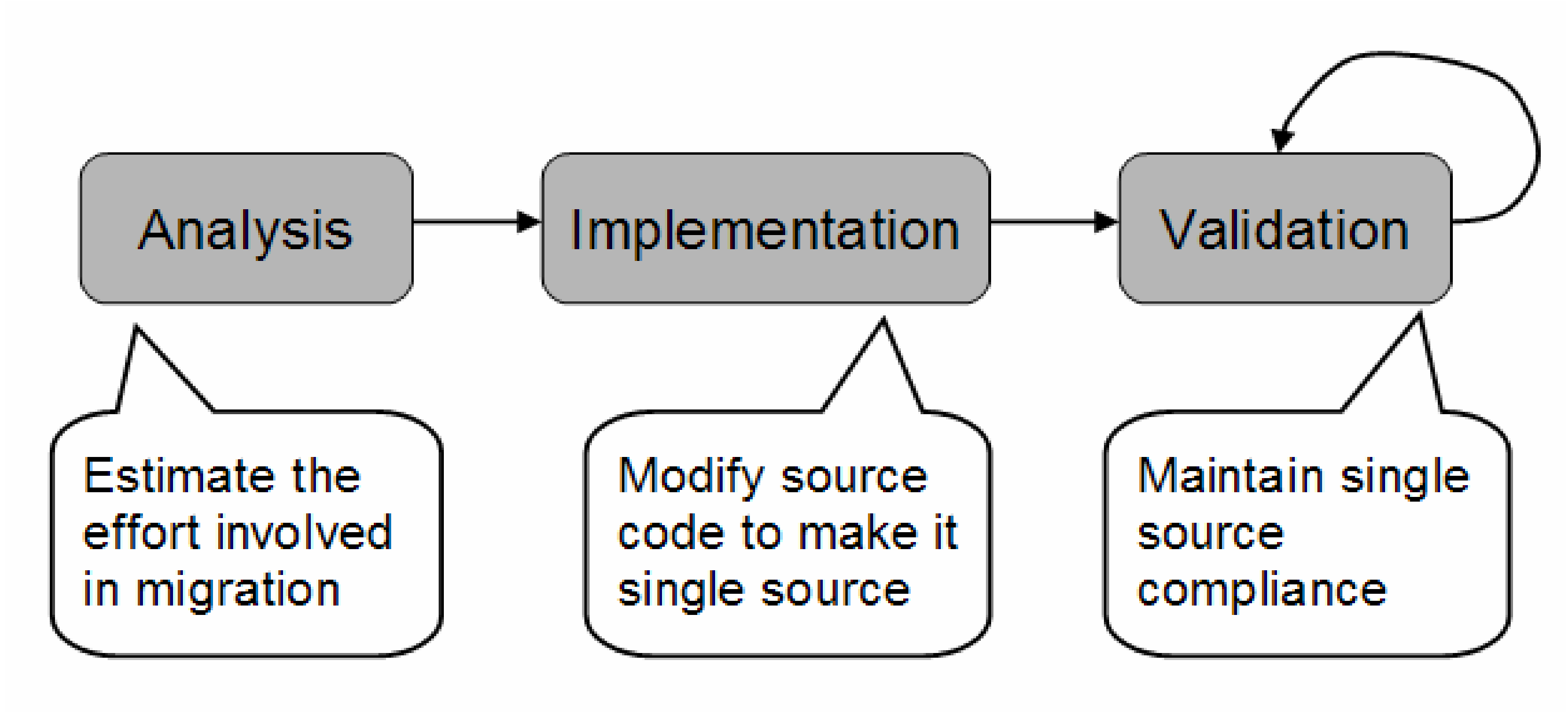


- Analyze a line or statement within a source file for migration errors
- Rules are created based on z/TPF migration guide and real customer migration experience
  - Example: TPF APAR PJ31999 and PJ32183 provide support for a packed decimal template class
    - TPF 4.1 C language applications that use z/OS `decimal(w,p)` data type should be **migrated** to use the decimal template class

TPF Toolkit provides 4 rules, **PJ32183a**, **PJ32183b**, **PJ32183c**, **PJ32183d**, to help with decimal type migration

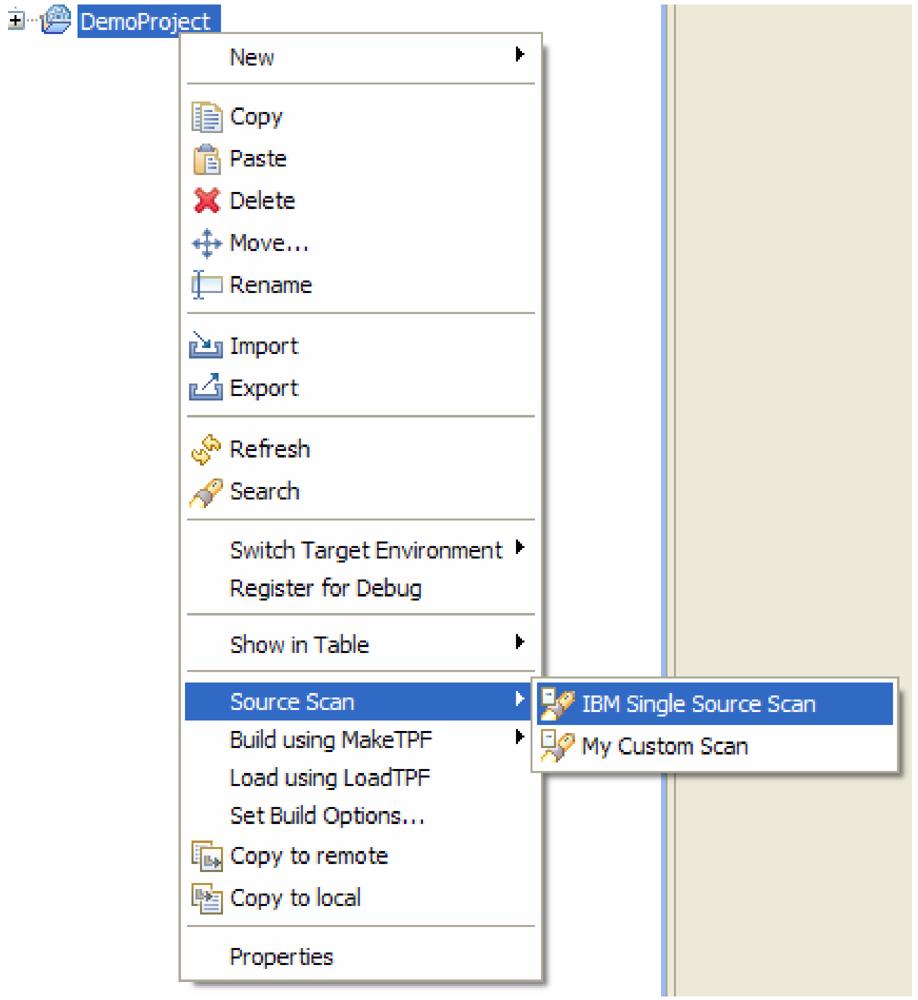
Source Scan

- **Three typical stages of a migration effort:**


Analysis
Implementation
Validation
Estimate the effort involved in migration
Modify source code to make it single source
Maintain single source compliance

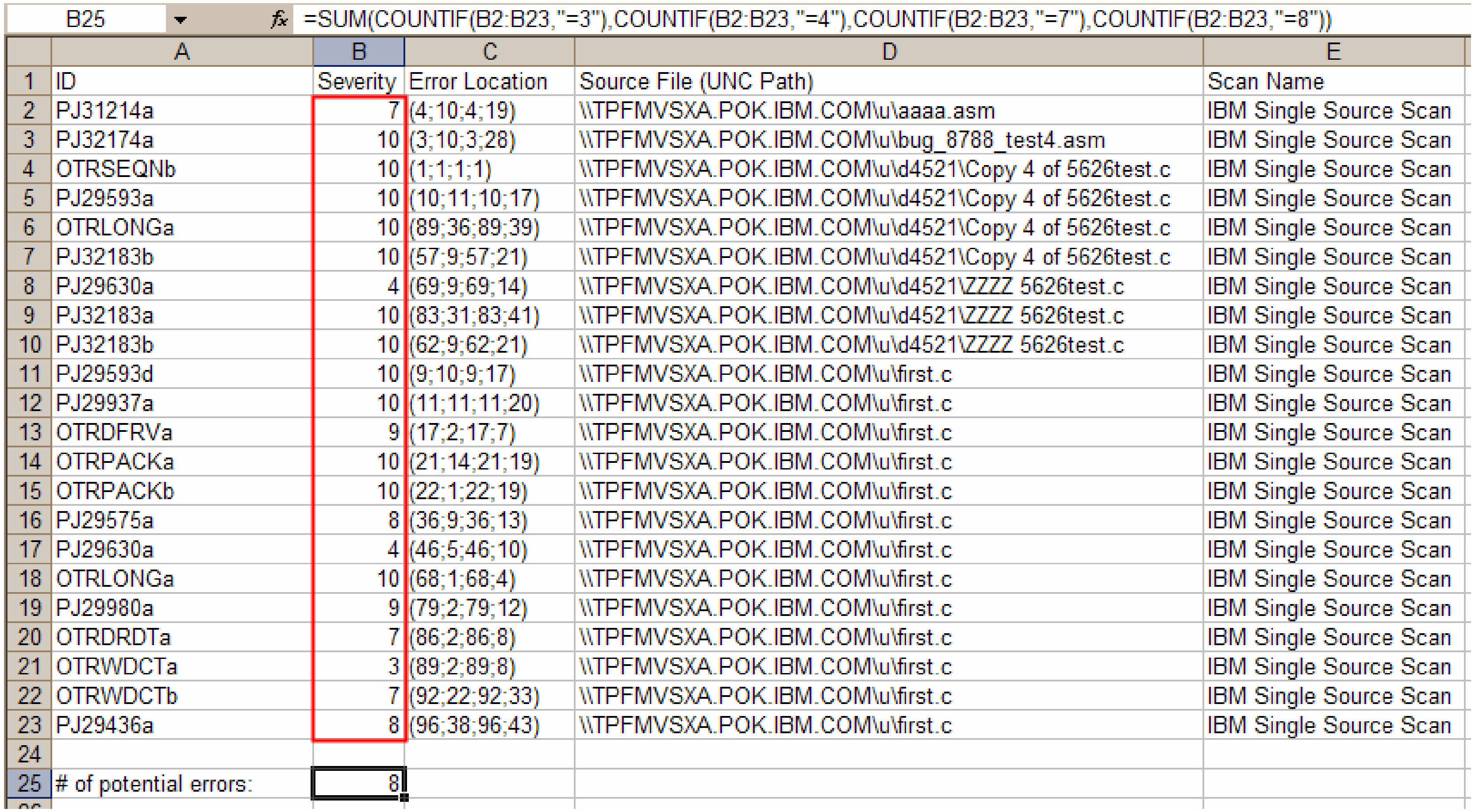
Analysis Phase

- **Scan your source code using built-in action and generate CSV errors file.**
- **TPF Toolkit provides command-line tool to scan entire directories and generate a CSV errors file**
  - Errors file can be analyzed to produce metrics
  - For example, CSV file can be imported into Microsoft Excel



Analysis Phase

- **Example Data**

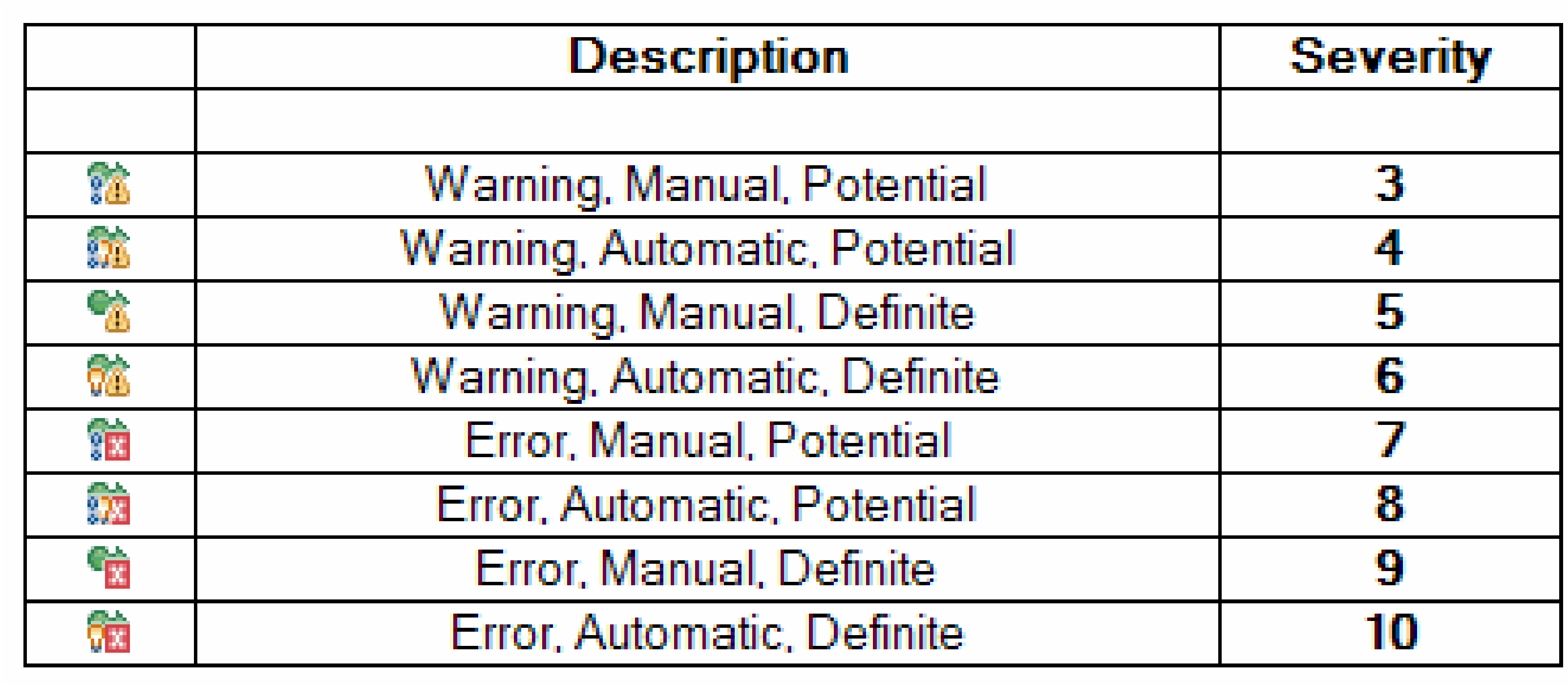


B25 =SUM(COUNTIF(B2:B23,"=3"),COUNTIF(B2:B23,"=4"),COUNTIF(B2:B23,"=7"),COUNTIF(B2:B23,"=8"))

	A	B	C	D	E
1	ID	Severity	Error Location	Source File (UNC Path)	Scan Name
2	PJ31214a	7	(4;10;4;19)	\\TPFMVSXA.POK.IBM.COM\u\aaaa.asm	IBM Single Source Scan
3	PJ32174a	10	(3;10;3;28)	\\TPFMVSXA.POK.IBM.COM\u\bug_8788_test4.asm	IBM Single Source Scan
4	OTRSEQNb	10	(1;1;1;1)	\\TPFMVSXA.POK.IBM.COM\u\d4521\Copy 4 of 5626test.c	IBM Single Source Scan
5	PJ29593a	10	(10;11;10;17)	\\TPFMVSXA.POK.IBM.COM\u\d4521\Copy 4 of 5626test.c	IBM Single Source Scan
6	OTRLONGa	10	(89;36;89;39)	\\TPFMVSXA.POK.IBM.COM\u\d4521\Copy 4 of 5626test.c	IBM Single Source Scan
7	PJ32183b	10	(57;9;57;21)	\\TPFMVSXA.POK.IBM.COM\u\d4521\Copy 4 of 5626test.c	IBM Single Source Scan
8	PJ29630a	4	(69;9;69;14)	\\TPFMVSXA.POK.IBM.COM\u\d4521\ZZZZ 5626test.c	IBM Single Source Scan
9	PJ32183a	10	(83;31;83;41)	\\TPFMVSXA.POK.IBM.COM\u\d4521\ZZZZ 5626test.c	IBM Single Source Scan
10	PJ32183b	10	(62;9;62;21)	\\TPFMVSXA.POK.IBM.COM\u\d4521\ZZZZ 5626test.c	IBM Single Source Scan
11	PJ29593d	10	(9;10;9;17)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
12	PJ29937a	10	(11;11;11;20)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
13	OTRDFRVa	9	(17;2;17;7)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
14	OTRPACKa	10	(21;14;21;19)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
15	OTRPACKb	10	(22;1;22;19)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
16	PJ29575a	8	(36;9;36;13)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
17	PJ29630a	4	(46;5;46;10)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
18	OTRLONGa	10	(68;1;68;4)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
19	PJ29980a	9	(79;2;79;12)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
20	OTRDRDTa	7	(86;2;86;8)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
21	OTRWDCTa	3	(89;2;89;8)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
22	OTRWDCTb	7	(92;22;92;33)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
23	PJ29436a	8	(96;38;96;43)	\\TPFMVSXA.POK.IBM.COM\u\first.c	IBM Single Source Scan
24					
25	# of potential errors:	8			

Analysis Phase

- **What do the severity numbers mean?**



	Description	Severity
	Warning, Manual, Potential	3
	Warning, Automatic, Potential	4
	Warning, Manual, Definite	5
	Warning, Automatic, Definite	6
	Error, Manual, Potential	7
	Error, Automatic, Potential	8
	Error, Manual, Definite	9
	Error, Automatic, Definite	10

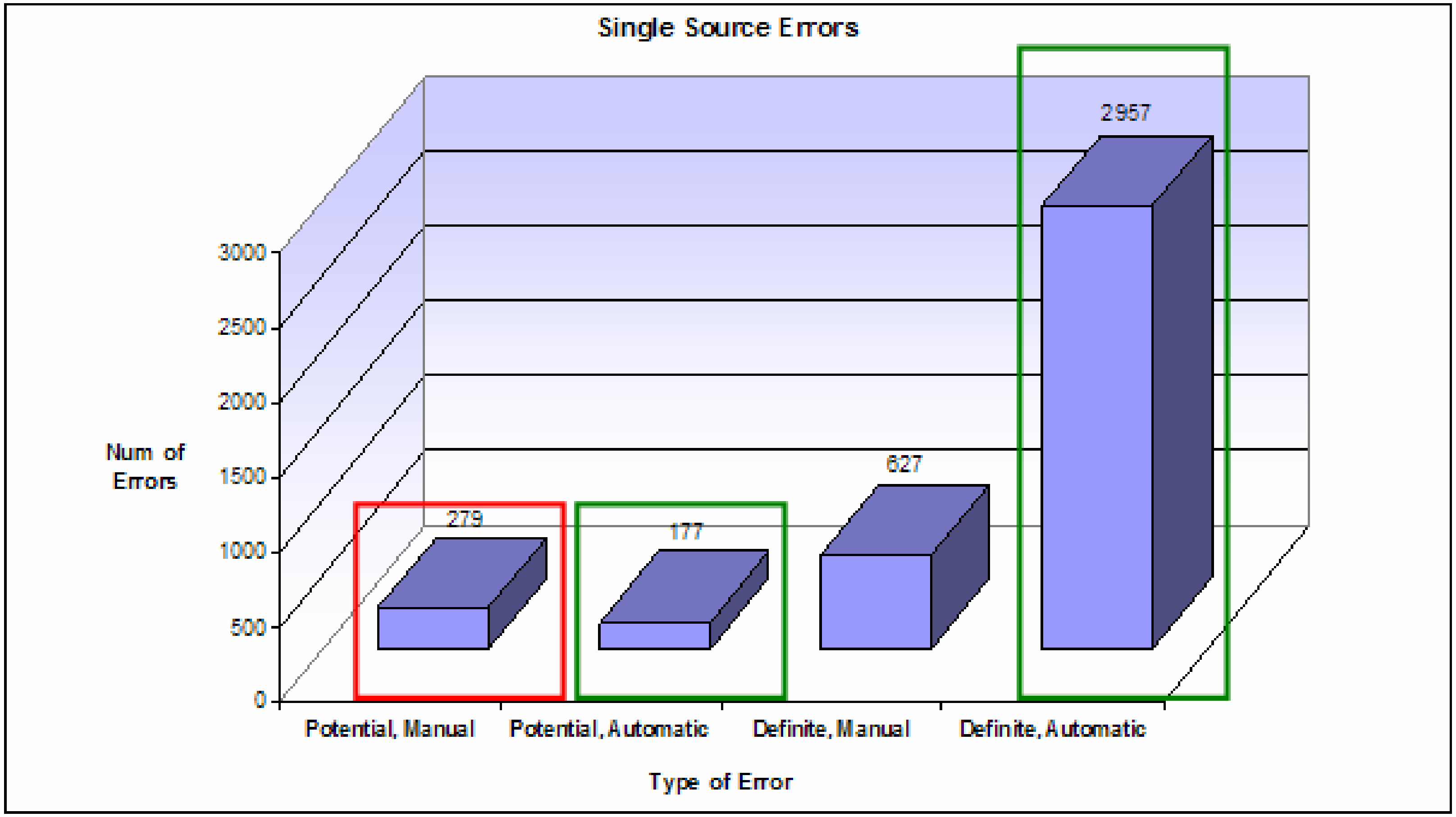
Analysis Phase

- **Types of errors flagged by rules: Definite vs. Potential**
  - Definite
    - Source Scan has enough information to determine that there is a **definite** migration problem
  - Potential
    - Source scan does not have enough information but there might be a **potential** migration problem
    - For example, lines in the source code might be a problem if certain runtime conditions are satisfied

Analysis Phase

- **Types of errors flagged by rules: Automatic vs. Manual fixes**
  - Automatic fix
    - Source Scan **can make the code change** for you to migrate your source code
  - Manual fix
    - Source scan **cannot make the code change** since additional information is needed

Analysis Phase

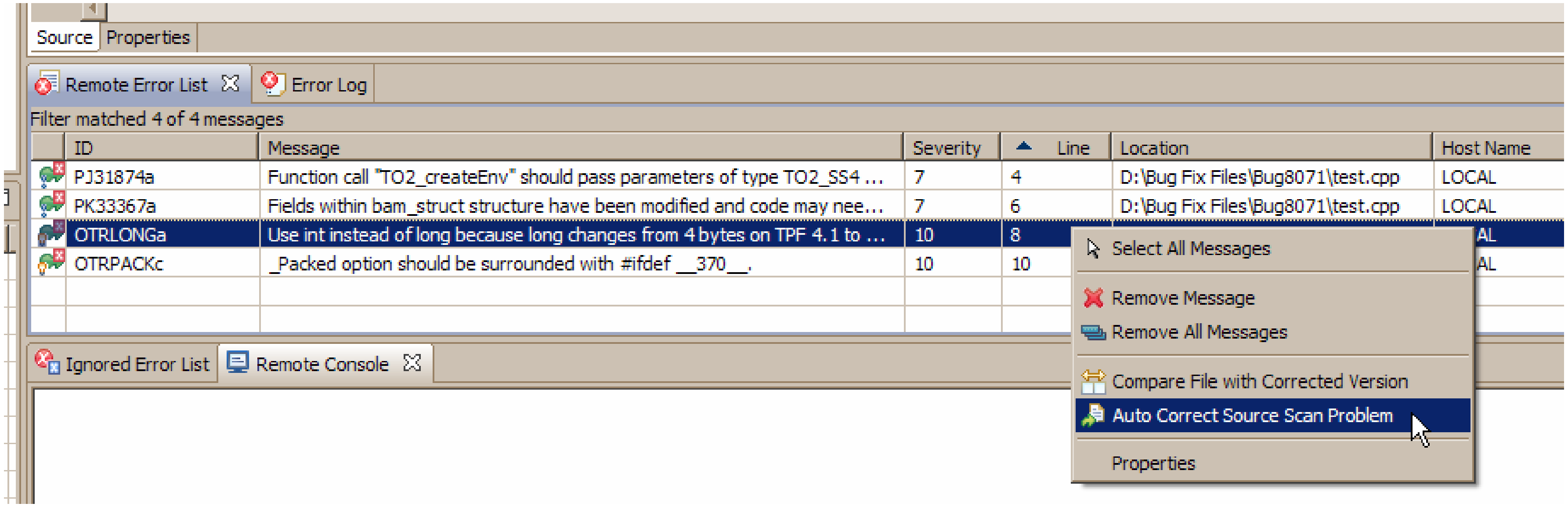


Analysis Phase

- **Estimate migration effort using certain metrics**
  - # of potential errors – Will require further investigation
  - # of errors with manual fixes – Will require modifications to source code by a developer

Implementation Phase: Automatic Fixes

- **Fix errors using TPF Toolkit**
  - *Auto Correct action* – Fix multiple errors automatically through the Remote Error List

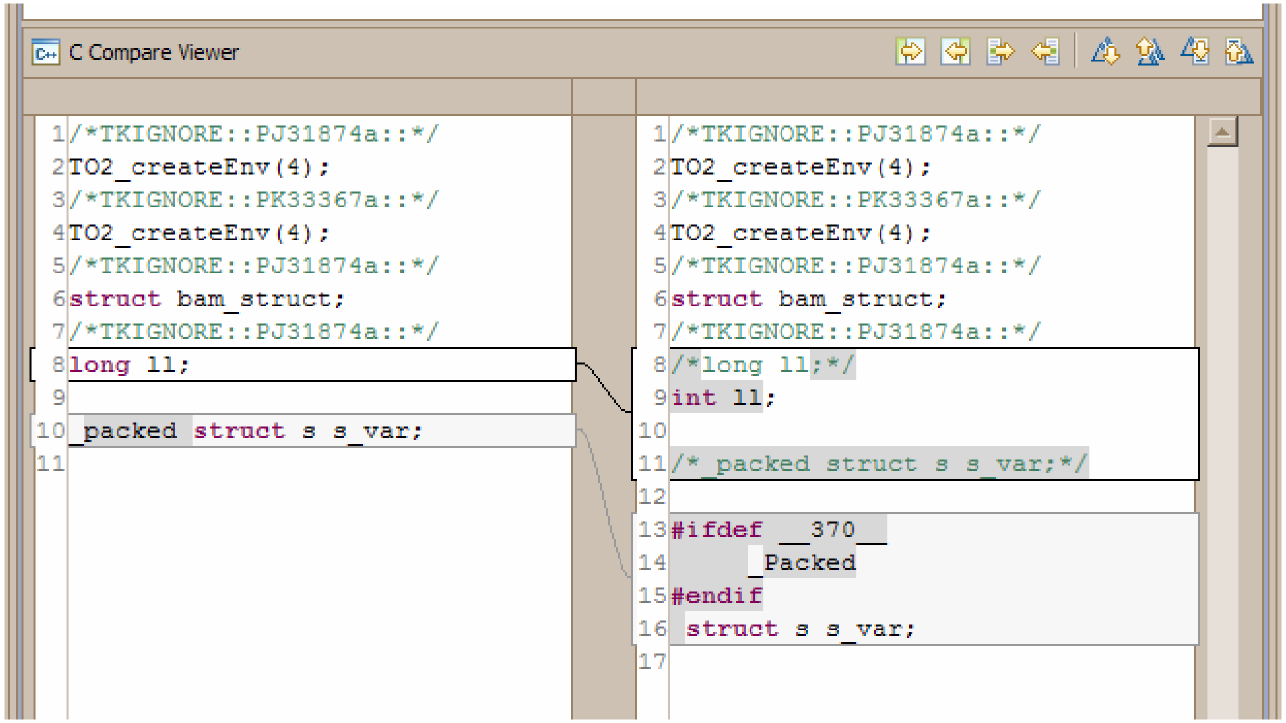


Filter matched 4 of 4 messages

ID	Message	Severity	Line	Location	Host Name
PJ31874a	Function call "TO2_createEnv" should pass parameters of type TO2_SS4 ...	7	4	D:\Bug Fix Files\Bug8071\test.cpp	LOCAL
PK33367a	Fields within bam_struct structure have been modified and code may nee...	7	6	D:\Bug Fix Files\Bug8071\test.cpp	LOCAL
OTRLONGa	Use int instead of long because long changes from 4 bytes on TPF 4.1 to ...	10	8		AL
OTRPACKc	_Packed option should be surrounded with #ifdef __370__.	10	10		AL

Implementation Phase: Automatic Fixes

- **Fix errors using TPF Toolkit**
  - *Compare editor* – Fix errors by comparing original source file with a corrected version



```
/*TKIGNORE::PJ31874a::*/
TO2_createEnv(4);
/*TKIGNORE::PK33367a::*/
TO2_createEnv(4);
/*TKIGNORE::PJ31874a::*/
struct bam_struct;
/*TKIGNORE::PJ31874a::*/
long ll;

_packed struct s s_var;
```

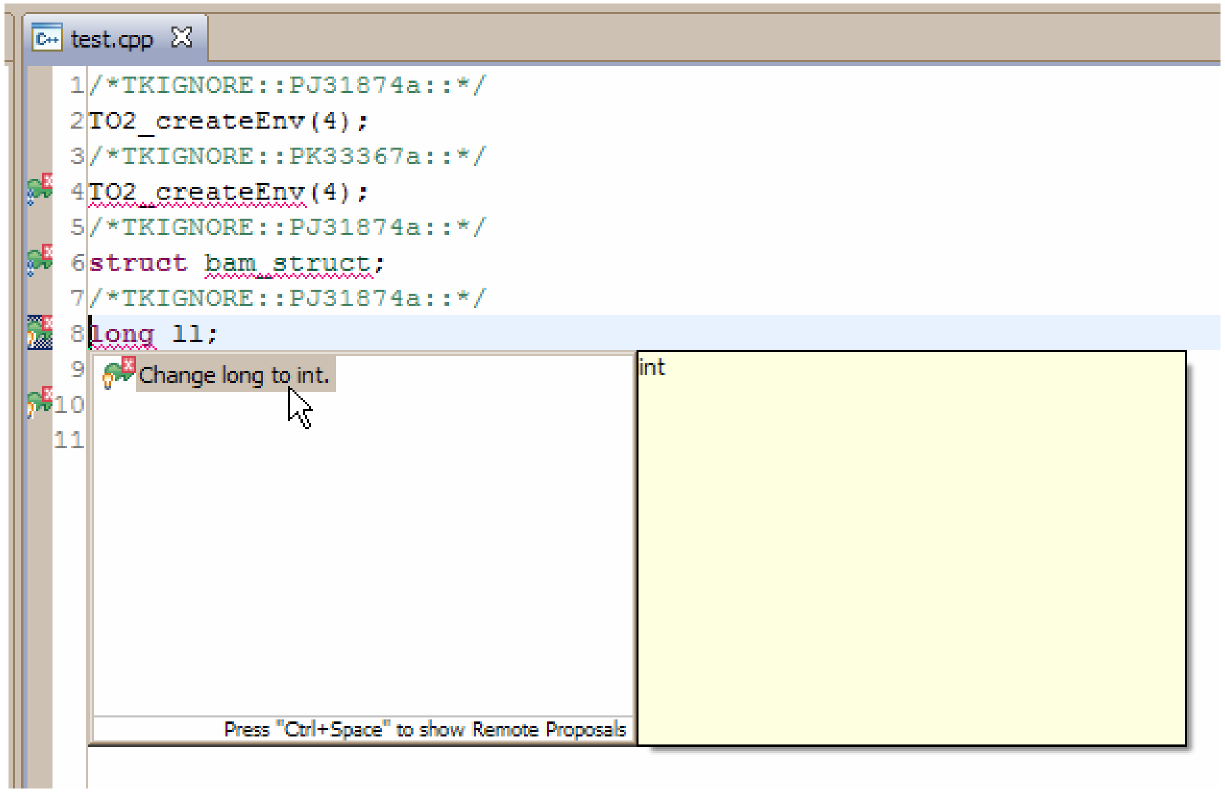
```
/*TKIGNORE::PJ31874a::*/
TO2_createEnv(4);
/*TKIGNORE::PK33367a::*/
TO2_createEnv(4);
/*TKIGNORE::PJ31874a::*/
struct bam_struct;
/*TKIGNORE::PJ31874a::*/
/*long ll;*/
int ll;

/*_packed struct s s_var;*/

#ifdef __370__
        _Packed
#endif
 struct s s_var;
```

Implementation Phase: Automatic Fixes

- **Fix errors using TPF Toolkit**
  - *Quick-fix action* – Fix individual errors through an editor

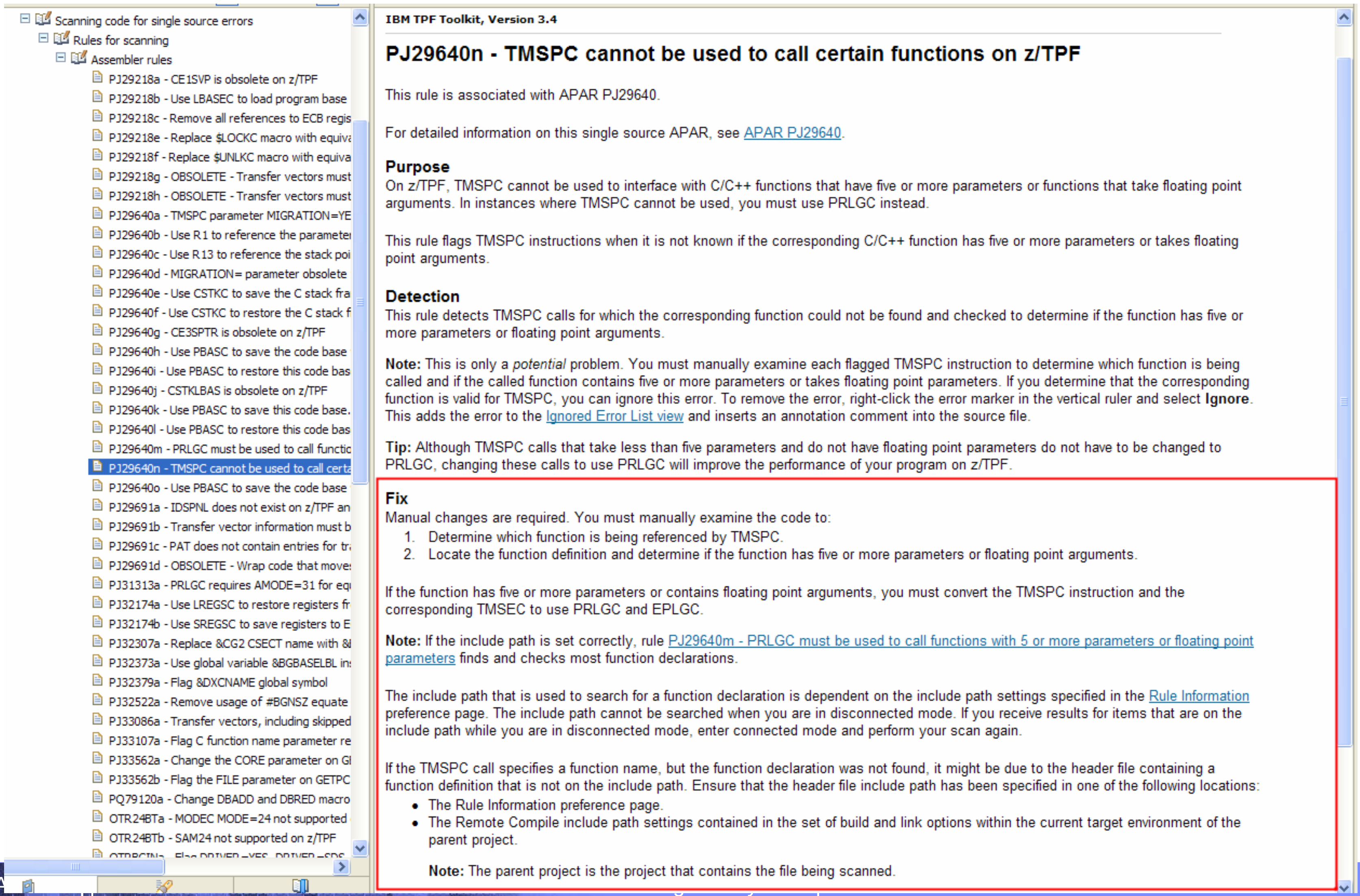


```
/*TKIGNORE::PJ31874a::*/
TO2_createEnv(4);
/*TKIGNORE::PK33367a::*/
TO2_createEnv(4);
/*TKIGNORE::PJ31874a::*/
struct bam_struct;
/*TKIGNORE::PJ31874a::*/
long ll;
```

Implementation Phase: Manual Fixes

- Developers need to investigate these errors
- A number of resources are available to assist
  - TPF Toolkit documentation
  - Information in the TPF Single Source APARs
  - z/TPF Migration Guide (http://www-306.ibm.com/software/htp/tpf/pages/gtpm1mst.pdf)

Implementation Phase: Manual Fixes



- Scanning code for single source errors
  - Rules for scanning
    - Assembler rules
      - PJ29218a - CE1SVP is obsolete on z/TPF
      - PJ29218b - Use LBASEC to load program base
      - PJ29218c - Remove all references to ECB regis
      - PJ29218e - Replace $LOCKC macro with equiva
      - PJ29218f - Replace $UNLKC macro with equiva
      - PJ29218g - OBSOLETE - Transfer vectors must
      - PJ29218h - OBSOLETE - Transfer vectors must
      - PJ29640a - TMSPC parameter MIGRATION=YE
      - PJ29640b - Use R1 to reference the paramete
      - PJ29640c - Use R13 to reference the stack poi
      - PJ29640d - MIGRATION= parameter obsolete
      - PJ29640e - Use CSTKC to save the C stack fra
      - PJ29640f - Use CSTKC to restore the C stack f
      - PJ29640g - CE3SPTR is obsolete on z/TPF
      - PJ29640h - Use PBASC to save the code base
      - PJ29640i - Use PBASC to restore this code bas
      - PJ29640j - CSTKLBAS is obsolete on z/TPF
      - PJ29640k - Use PBASC to save this code base.
      - PJ29640l - Use PBASC to restore this code bas
      - PJ29640m - PRLGC must be used to call functic
      - PJ29640n - TMSPC cannot be used to call certa
      - PJ29640o - Use PBASC to save the code base
      - PJ29691a - IDSPNL does not exist on z/TPF an
      - PJ29691b - Transfer vector information must b
      - PJ29691c - PAT does not contain entries for tr
      - PJ29691d - OBSOLETE - Wrap code that move
      - PJ31313a - PRLGC requires AMODE=31 for eq
      - PJ32174a - Use LREGSC to restore registers fr
      - PJ32174b - Use SREGSC to save registers to E
      - PJ32307a - Replace &CG2 CSECT name with &
      - PJ32373a - Use global variable &BGBASELBL in
      - PJ32379a - Flag &DXCNAME global symbol
      - PJ32522a - Remove usage of #BGNSZ equate
      - PJ33086a - Transfer vectors, including skipped
      - PJ33107a - Flag C function name parameter re
      - PJ33562a - Change the CORE parameter on GI
      - PJ33562b - Flag the FILE parameter on GETPC
      - PQ79120a - Change DBADD and DBRED macro
      - OTR24BTa - MODEC MODE=24 not supported
      - OTR24BTb - SAM24 not supported on z/TPF

IBM TPF Toolkit, Version 3.4

PJ29640n - TMSPC cannot be used to call certain functions on z/TPF

This rule is associated with APAR PJ29640.

For detailed information on this single source APAR, see APAR PJ29640.

Purpose
On z/TPF, TMSPC cannot be used to interface with C/C++ functions that have five or more parameters or functions that take floating point arguments. In instances where TMSPC cannot be used, you must use PRLGC instead.

This rule flags TMSPC instructions when it is not known if the corresponding C/C++ function has five or more parameters or takes floating point arguments.

Detection
This rule detects TMSPC calls for which the corresponding function could not be found and checked to determine if the function has five or more parameters or floating point arguments.

Note: This is only a *potential* problem. You must manually examine each flagged TMSPC instruction to determine which function is being called and if the called function contains five or more parameters or takes floating point parameters. If you determine that the corresponding function is valid for TMSPC, you can ignore this error. To remove the error, right-click the error marker in the vertical ruler and select **Ignore**. This adds the error to the Ignored Error List view and inserts an annotation comment into the source file.

Tip: Although TMSPC calls that take less than five parameters and do not have floating point parameters do not have to be changed to PRLGC, changing these calls to use PRLGC will improve the performance of your program on z/TPF.

Fix
Manual changes are required. You must manually examine the code to:

1. Determine which function is being referenced by TMSPC.
2. Locate the function definition and determine if the function has five or more parameters or floating point arguments.

If the function has five or more parameters or contains floating point arguments, you must convert the TMSPC instruction and the corresponding TMSEC to use PRLGC and EPLGC.

Note: If the include path is set correctly, rule PJ29640m - PRLGC must be used to call functions with 5 or more parameters or floating point parameters finds and checks most function declarations.

The include path that is used to search for a function declaration is dependent on the include path settings specified in the Rule Information preference page. The include path cannot be searched when you are in disconnected mode. If you receive results for items that are on the include path while you are in disconnected mode, enter connected mode and perform your scan again.

If the TMSPC call specifies a function name, but the function declaration was not found, it might be due to the header file containing a function definition that is not on the include path. Ensure that the header file include path has been specified in one of the following locations:

- The Rule Information preference page.
- The Remote Compile include path settings contained in the set of build and link options within the current target environment of the parent project.

  Note: The parent project is the project that contains the file being scanned.

Validation Phase

- **Ensure that migrated code remains single source compliant**

- **Command-line tool to scan source directories**
  - Schedule nightly scans of source code

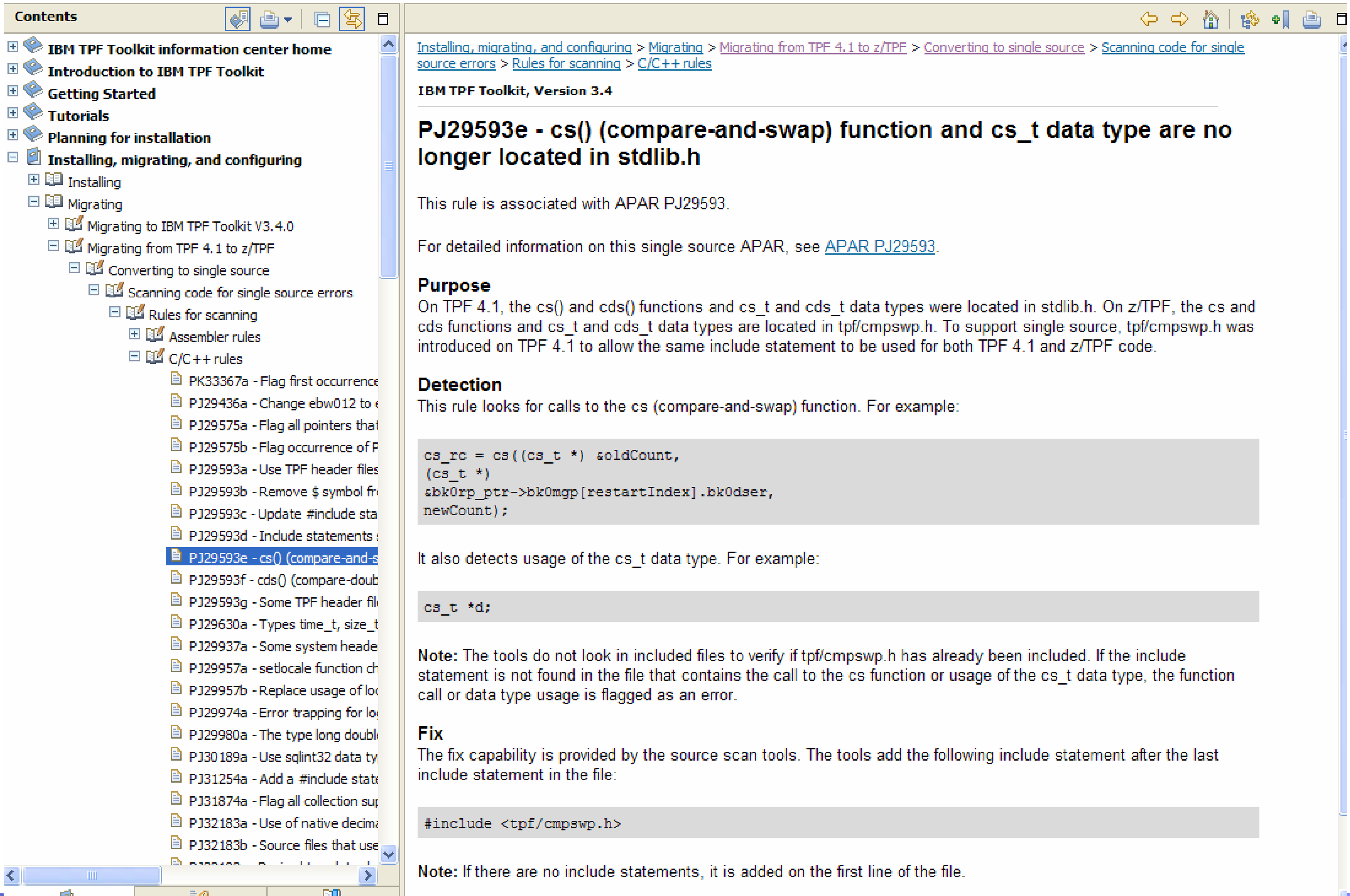
Validation Phase

- **Validation when files are edited by users**
  - Errors are automatically flagged when file is saved
- **Validation using custom enterprise actions**
  - E.g. Create a source check-in action which only proceeds if no errors are present in source file

Resources

- **Documentation within TPF Toolkit**
    - Installing, migrating, and configuring > Configuring > Source scan
    - Installing, migrating, and configuring > Migrating > Migrating from TPF 4.1 to z/TPF

Resources



Contents

- IBM TPF Toolkit information center home
- Introduction to IBM TPF Toolkit
- Getting Started
- Tutorials
- Planning for installation
- Installing, migrating, and configuring
  - Installing
  - Migrating
    - Migrating to IBM TPF Toolkit V3.4.0
    - Migrating from TPF 4.1 to z/TPF
      - Converting to single source
        - Scanning code for single source errors
          - Rules for scanning
            - Assembler rules
            - C/C++ rules
              - PK33367a - Flag first occurrence
              - PJ29436a - Change ebw012 to e
              - PJ29575a - Flag all pointers that
              - PJ29575b - Flag occurrence of P
              - PJ29593a - Use TPF header files
              - PJ29593b - Remove $ symbol fr
              - PJ29593c - Update #include sta
              - PJ29593d - Include statements
              - PJ29593e - cs() (compare-and-s
              - PJ29593f - cds() (compare-doub
              - PJ29593g - Some TPF header fil
              - PJ29630a - Types time_t, size_t
              - PJ29937a - Some system heade
              - PJ29957a - setlocale function ch
              - PJ29957b - Replace usage of loc
              - PJ29974a - Error trapping for lo
              - PJ29980a - The type long doubl
              - PJ30189a - Use sqlint32 data ty
              - PJ31254a - Add a #include state
              - PJ31874a - Flag all collection sup
              - PJ32183a - Use of native decima
              - PJ32183b - Source files that use

Installing, migrating, and configuring > Migrating > Migrating from TPF 4.1 to z/TPF > Converting to single source > Scanning code for single source errors > Rules for scanning > C/C++ rules

IBM TPF Toolkit, Version 3.4

PJ29593e - cs() (compare-and-swap) function and cs_t data type are no longer located in stdlib.h

This rule is associated with APAR PJ29593.

For detailed information on this single source APAR, see APAR PJ29593.

Purpose

On TPF 4.1, the cs() and cds() functions and cs_t and cds_t data types were located in stdlib.h. On z/TPF, the cs and cds functions and cs_t and cds_t data types are located in tpf/cmpswp.h. To support single source, tpf/cmpswp.h was introduced on TPF 4.1 to allow the same include statement to be used for both TPF 4.1 and z/TPF code.

Detection

This rule looks for calls to the cs (compare-and-swap) function. For example:

```
cs_rc = cs((cs_t *) &oldCount,
(cs_t *)
&bk0rp_ptr->bk0mgp[restartIndex].bk0dser,
newCount);
```

It also detects usage of the cs_t data type. For example:

```
cs_t *d;
```

Note: The tools do not look in included files to verify if tpf/cmpswp.h has already been included. If the include statement is not found in the file that contains the call to the cs function or usage of the cs_t data type, the function call or data type usage is flagged as an error.

Fix

The fix capability is provided by the source scan tools. The tools add the following include statement after the last include statement in the file:

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

Note: If there are no include statements, it is added on the first line of the file.

Resources

- **Previous TPFUG presentations**
- **“Migrating to z/TPF using TPF Toolkit” education session on Wednesday**

Questions

Trademarks