



TSI



MOVE FORWARD · TOGETHER

# Product Updates

## TPFUG 2019, Denver



# Agenda

- zTPFGI Releases
  - ▶ JavaNow
  - ▶ Multiple System Support
  - ▶ DFDL with Rest API
  - ▶ zQDC Automation
  - ▶ Dump Viewer – Source, TPFDF, DETAC
  - ▶ Color Customization
  - ▶ Other Enhancements

# zTPFGI Releases

zTPFGI continues to deliver 2 releases per year, focused on priority customer requests, fixes, and the growing feature set of z/TPF®

- ▶ This presentation covers features from three different zTPFGI releases:
  - ▶ **1.3.8.0**
    - › Being rolled out at customers now
  - ▶ **1.3.9.0**
    - › Testing for rollout soon
  - ▶ **1.4.0.0**
    - › Scheduled for release this summer
- ▶ Unless otherwise marked, features belong to 1.3.8.0
- ▶ Other features will be marked with a \* for 1.3.9.0 and a \*\* for 1.4.0.0

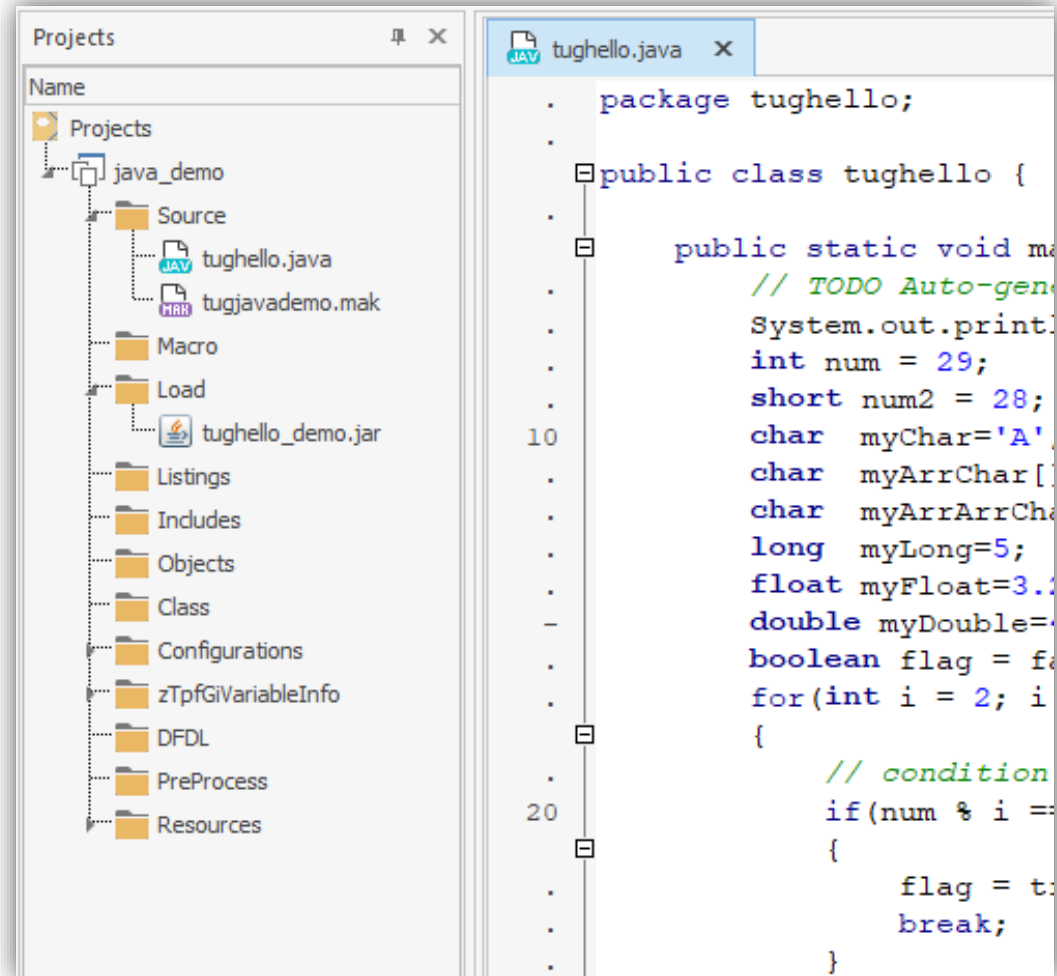
## What is JavaNow?

- ▶ z/TPF shops' workflow will include a Java-specific IDE for Java resources (e.g. NetBeans, IntelliJ, Eclipse), and...
- ▶ **JavaNow** in zTPFGI is an additional option that allows Java for z/TPF to be coded, compiled, loaded and debugged directly in zTPFGI

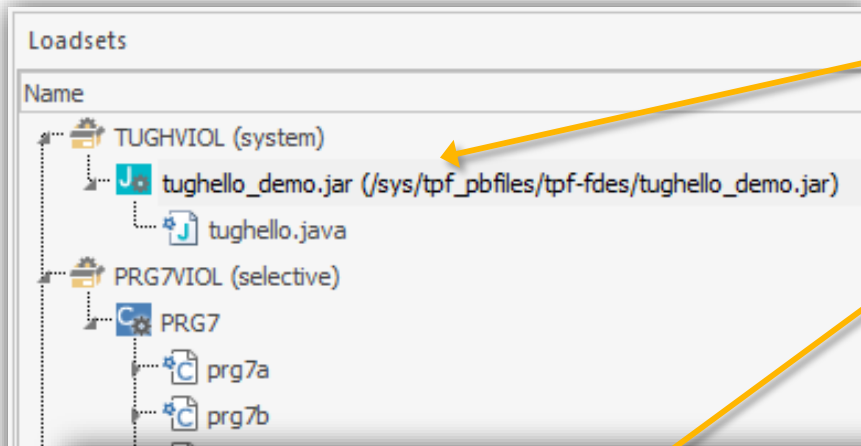
## JavaNow Benefits

- ▶ Saves **time** for developers who are primarily interested in traditional z/TPF development but who need to interact with Java for development, testing, debugging, and maintenance
- ▶ Saves **training costs** for developers who know zTPFGI and also need to work with Java

- ▶ Support for editing Java files (syntax highlighting)
- ▶ Support for Java files in projects
- ▶ Support for compile/build of Java
- ▶ Support for Loading Jar files

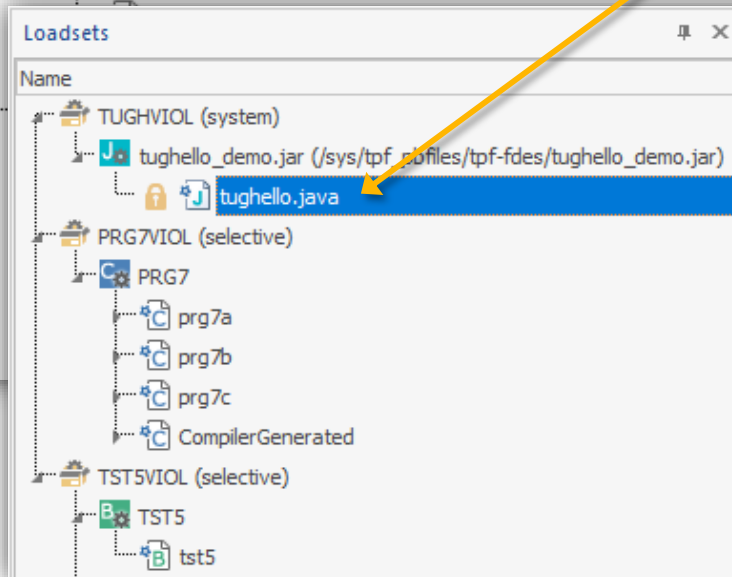


# JavaNow Loadsets View



▶ Expand jar files in Loadsets View to display Java files

▶ Double click to lock Java files for Source View Trace



```
tughello.java x
package tughello;

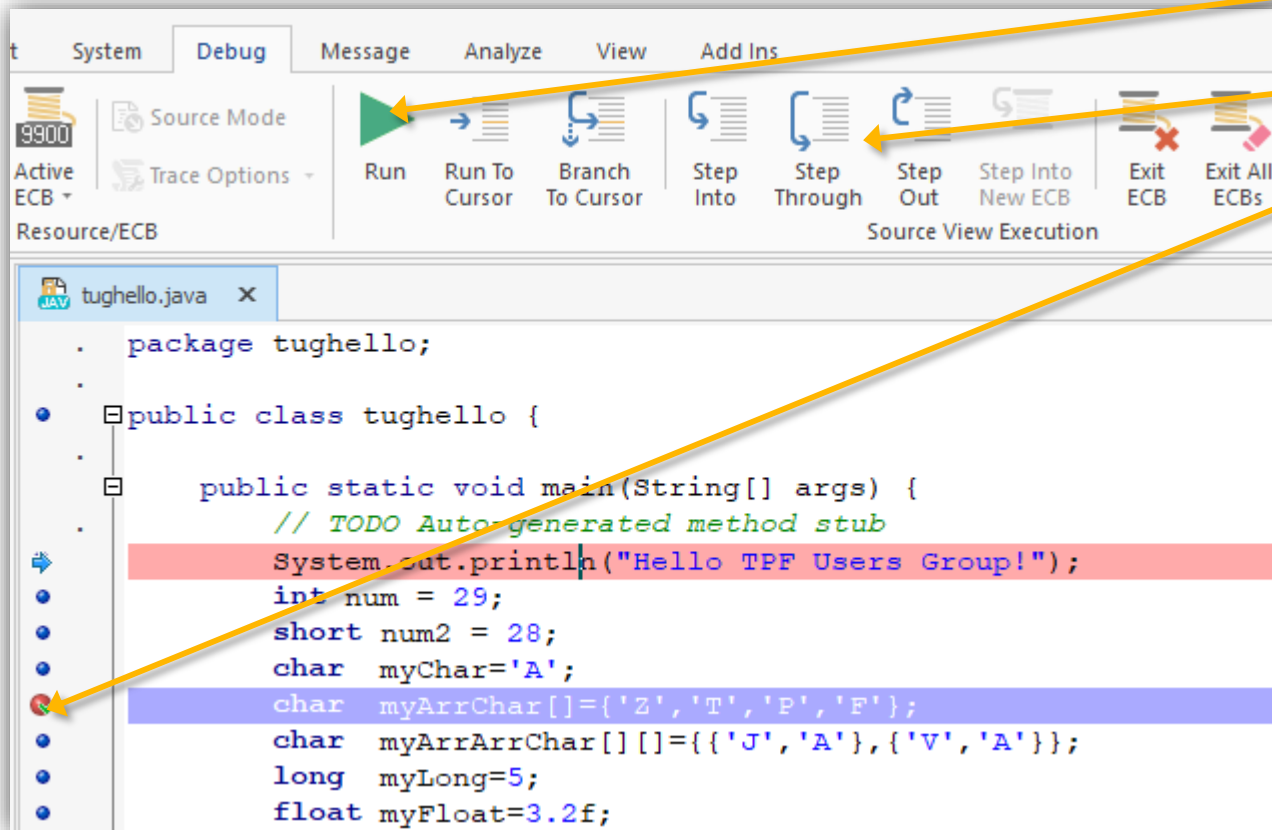
public class tughello {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello TPF Users");
        int num = 29;
        short num2 = 28;
        char myChar='A';
        char myArrChar[]={ 'Z', 'T', 'P', 'F' };
        char myArrArrChar[][]={{ 'J', 'A' }, { 'V', 'I' }};
        long myLong=5;
        float myFloat=3.2f;
        double mvDouble=4.5d;
    }
}
```

# JavaNow Source View Trace Execution



Source View Trace Java in same execution and control buttons as Assembler/C/C++/SabreTalk



- ▶ Run
- ▶ Step
- ▶ Set Breakpoints
- ▶ Etc.

# JavaNow Source View Trace Variables



Watches

Name	Value
num	29

Call Stack

Function	Member	SO	Line	Stack Address
tughello.tughello.main	tughello.java	tugh	21	000000000000001000

Variables

Name	Value
i	2
num	29
num2	28

Source View Trace Java with same views as Assembler/ C/C++/SabreTalk

- ▶ Watches View
- ▶ Call Stack View
- ▶ Variables View
- ▶ Hover over code for variable values

```
long myLong=5;
float myFloat=3.2f;
double myDouble=4.5d;
boolean flag = false;
for(int i = 2; i <= num/2; ++i)
{
    // condition for nonprime number
    if(num % i == 0)
    {
        break;
    }
}

if (!flag)
    System.out.println(num + " is a prime number.");
```



# JavaNow Demo



The screenshot displays the JavaNow IDE interface. The main editor window shows the source code for `tughello.java` with the following content:

```
package tughello;

public class tughello {

    public static void main(String[] args) {
        int num = 29;
        short num2 = 28;
        char myChar = 'A';
        char myArrChar[] = {'z','T','P','F'};
        char myArrArrChar[][] = {'J','A'},{'V','A'};
        long myLong = 5;
        float myFloat = 3.2f;
        double myDouble = 4.5d;
        boolean flag = false;

        System.out.println("Hello TPFUG!");

        // is num prime?
        for(int i = 2; i <= num/2; ++i)
        {
            // condition for nonprime number
            if(num % i == 0)
            {
                // ...
            }
        }
    }
}
```

The interface includes a menu bar (File, Home, Source, Page Layout, System, Message, Analyze, View, Add Ins), a toolbar with various tool icons, and a sidebar with a Loadsets table and a Projects tree. The Projects tree shows a project named `java_demo` with sub-projects like `Source`, `Macro`, `Load`, `Listings`, `Includes`, `Objects`, `Class`, `Configurations`, `zTpfGivVariableInfo`, `DFDL`, and `PreProcess`. The `Source` sub-project contains `tughello.java` and `tugjavademo.mak`. The Output window at the bottom shows the following log messages:

```
15:46:58 Connecting to system VPZTPPF4
15:46:58 Connected to system VPZTPPF4
```

The status bar at the bottom indicates the system is ready and provides details like `Total Pool F/A's - Gets:0, Rel:0; Details : OFF` and `CAPS NUM SCRL Insert 100%`.

# Multiple System Support\*

Multi-Sys

## What is it?

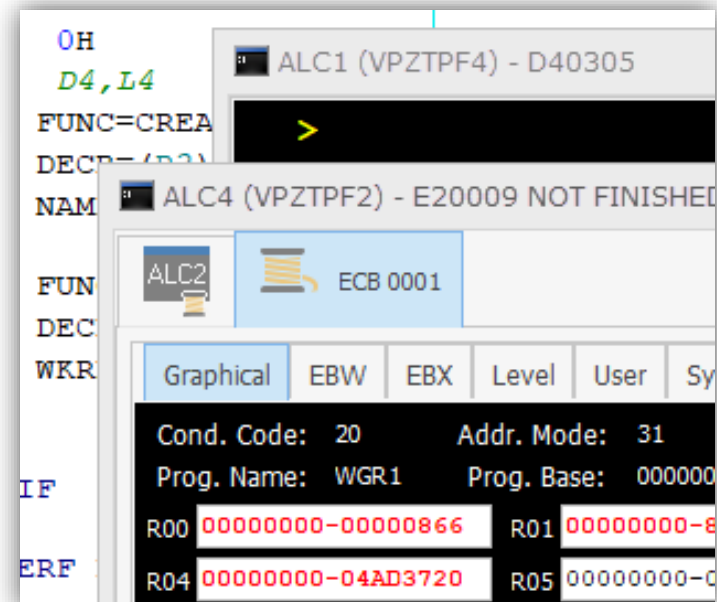
- ▶ zTPFGI now supports connections to multiple system/CPU configurations from a single instance of zTPFGI. (Eliminates the need for multiple instances of zTPFGI.)

## Benefits

- ▶ Much quicker to set up for debugging multiple systems and loosely coupled systems

## How it works

- ▶ Connecting to multiple systems is one step, the same as connecting to a single system
- ▶ Multiple system configurations are set up by zTPFGI administrator
- ▶ The system being targeted is identified on each zTPFGI window/view



zTPFGI's existing DFDL support includes editing, testing and loading of DFDL schemas

The screenshot displays the zTPFGI IDE interface for editing a DFDL schema. The main window shows a tree view of the schema structure for 'GR3MSR.tpddf.dfdl.xsd'. The 'Global Elements' section is expanded, showing a table of elements:

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
xs:sequence		1	1		
IndexName1	ns0:IndexName1	0	1		
xs:sequence		1	1		
PrefixdLREC	ns0:PrefixdLREC				
xs:sequence		1	1		
LRECPrefix		1	1		
xs:sequence		1	1		
DFLREC	ns0:DFLREC	1	1		

The 'DFDL Properties' panel on the right shows the configuration for the selected 'IndexName1' element:

DFDL Properties	
IndexName1 (xs:element)	
<b>General</b>	
Name	IndexName1
Type	ns0:IndexName1
Data Format Reference	< default data form
Encoding	ebcdic-cp-us
Byte Order	bigEndian
Ignore Case	no
Fill Byte	0
<b>Content</b>	
Length Kind	explicit
Length	8
Length Units	bytes
Niltable	false
<b>Occurrences</b>	
Min Occurs	0
Max Occurs	1
<b>Alignment</b>	
Alignment	1
Alignment Units	bytes
Leading Skip	0
Trailing Skip	0

The 'Output' window at the bottom shows a schema definition error:

```
GR3MSR.tpddf.dfdl.xsd /home/ed/ape4/dfd1_tpddf/
Line 17: 18 "Schema Definition Error: occursCountKind='fixed' requires minOccurs and maxOccurs to be equal (0 != 1)"
```

## Need

- ▶ z/TPF systems need to provide services to be consumed through Rest API

## What is TPF Rest Services Tool?

- ▶ The TPF Rest Services Tool is integrated with zTPFGI
- ▶ It allows users to build the files (service JSON files, swagger schemas and JAM descriptors) needed to make API communication within or outside the TPF system

## Benefits

- ▶ All that is needed to create a service is available in a tool in zTPFGI
- ▶ Uses standard files. Users can edit and reuse product files created by other tools.
- ▶ Simple to use with immediate feedback, saving time

# TPF Rest Services – Services Artifact

DFDL/Rest

- ▶ Allows the user to generate the service descriptor (Service JSON) file that describes the implementation for z/TPF services
- ▶ Generates based on DFDL schemas that are easy to select in the zTPFGL environment

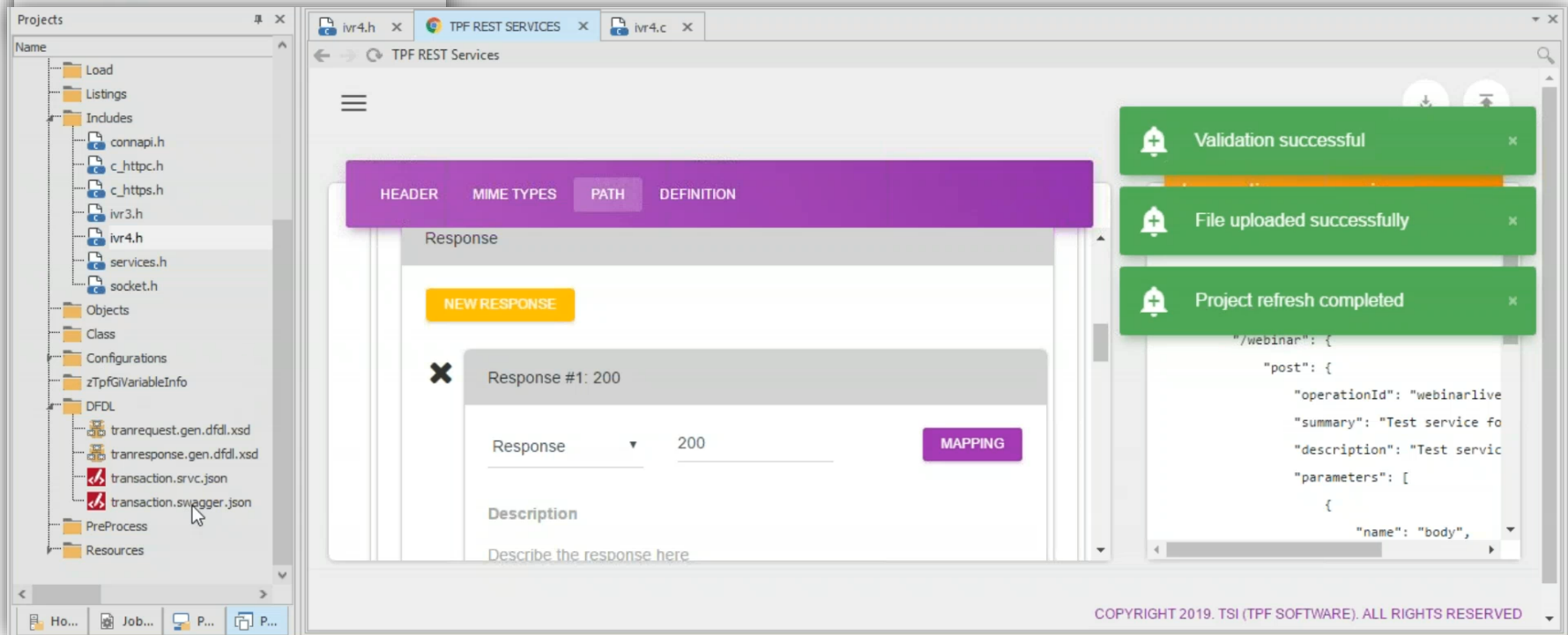
The screenshot shows the TSI TPF REST Services web application. The interface includes a left sidebar with the TSI logo and navigation options: Service Artifacts, Swagger Generation, and JAM Descriptor. The main content area is titled "Service JSON Generation" and features a "Response Format" section. In this section, a file named "tranresponse\_gen.dfdl.xsd" is selected, with a "CHANGE DFDL" button next to it. Below this, a "Root" dropdown menu is set to "tranresponse". On the right side, a preview window displays the generated "transaction.srvc.json" file, showing a JSON structure with fields like "version", "operationId", "description", "providerType", "provider", "request", and "response". The footer of the application contains the text "COPYRIGHT 2019. TSI (TPF SOFTWARE). ALL RIGHTS RESERVED" and a status bar with "CAPS NUM SCRL INS Insert 100%" and zoom controls.

# TPF Rest Services – Swagger

DFDL/Rest



- ▶ Allows the user to generate OpenAPI descriptor (Swagger) that describes a set of Rest APIs
- ▶ The swagger JSON is generated based on a selected service JSON file



# TPF Rest Services – JAM Descriptor

DFDL/Rest

- ▶ Allows the user to generate JAM descriptor that contains the necessary information to define JVMs for Java applications on z/TPF
- ▶ The JAM.XML file is generated based on the service JSON and can be loaded to the development server
- ▶ Then the JAM can be uploaded to the z/TPF system to make API communication

The screenshot displays the TSI JAM Descriptor Wizard interface. The sidebar on the left includes the TSI logo and navigation options: Service Artifacts, Swagger Generation, and JAM Descriptor. The main content area is titled "Jam Descriptor Wizard Details" and contains the following fields:

- JAM Name\***: Flight
- JAM Description**: (Empty field)
- Path Details\***: /sys/tpf\_pbfiles/
- Other Command Line Options**: (Empty field)

On the right, a preview window shows the generated .jam.xml file content:

```
<?xml version='1.0' encoding='UTF-8'? >
<tns: JAM xmlns:tns='http://www.ibm.com/xm
xmlns: xsi = 'http://www.w3.org/2001/XMLSc
  <tns:JAMName>Flight</tns:JAMName>
  <tns:JVMCommandLineOptions>
    <tns:ClassPath>/sys/tpf_pb
  </tns:JVMCommandLineOptions>
</tns:JAM>
```

# Dump Viewer\*

## View Dump Source

- ▶ Option to open read-only source file, if available, showing execution line where dump occurred

The screenshot displays the Dump Viewer application interface. The main window shows the source code for `dmpv.asm` with the following assembly instructions:

```
200 GETCC , DECB
    LGHI R4, 100
    LA R5, 1
    CALOC COUNT=
    **** MK ****
    LARL R3, =X'
    LG R5, 0(R
    MVC EBX00
    SPACE
    J ERREND
210 *****
    * CASE FOR SERRC WITH
    *****
    ERRCAS5 DS 0H
    SERRC E, 01234
    J ERREND
    *****
```

The instruction `MVC EBX00` is highlighted in red, indicating the location of the dump. The Dump Viewer window is open, showing the following details:

- Cond. Code: 00, Addr. Mode: 64, Prot. Key: 00
- Prog. Name: DMPV, Prog. Base: 00000000947D1000
- Registers: R00 (00000000-000000FF), R01 (00000000-12464E80), R02 (00000000-000003E8), R05 (0FFFFFFF-FDFFFDFF), R06 (00000000-947D1078), R09 (00000000-12400000), R10 (00000000-00002000), R13 (00000000-947D160E), R14
- Data Levels: D0 (S), D1 (H), D2, D3 (H), D4, D5, D8, D9, DA, DB, DC, DD
- FA0: 00000000 00000000 .. CR0: 12464E80 0021017D
- DECBs: IDFSLEVELDD@, IDFFGR31SR . (S), IDFFGR31SR ., IDFFGR30SR
- Dump Details: Dump Date: 27MAR, Dump Time: 14.25.10, Dump Number: 00000004, CPU ID: B, Terminal:



# Dump Viewer

## View Dump Source

- ▶ Option to open read-only source file, if available, showing execution line where dump occurred.

## View TPFDF in DF Insight

- ▶ Click on DF Insight in the Dump Viewer to view TPFDF just as you would when tracing an ECB.

DF Insight DMP1 9947

Name	SW00SR	Data	Space
GR31SR1			
GR25SR			
GR28SR			
GR31SR.4			
GR30SR			
GR32SR			

### GR32SR

Property	Value
ACTIVE FILE REF	GR32SR
SW00SR ADDRESS	0000000017A0DD00
SW00SR SIZE	00000000
SW00CMD 2C8 Command	SPACE
SW00PGM 004 Last Program	DMPV
SW00WID 014 FILEID	FDFD
SW00INB 018 Initial NAB	001A
SW00ILT 01C Fixed Rec Len	0001
SW00OP1 02B Option Byte 1	80
	Full Backward Chaining Wanted
SW00OP2 02C option Byte 2	06
	Use HOLD for Chain Retrieval
	DUMP DB010C if HOLD VIOLATION
SW00ORD 0A4 Ordinal	00000000
SW00FAD8 050 File Address	00000000CC006000
SW00IPT8 048 NEWLREC Address	0000000000000000
DATA----->	00000000000000000000000000000000
SW00RIN 080 Return Code	40
	40 Wanted item not found
SW00REC 094 Located LREC	00000000
SW00NKY 09E Key Type	00NKYTEXT
SW00AL# 1C0 Detac Count	0000
SW00PCA 1E8 Prime Block	12416BEO
SW00CAS 194 Current Block	0000000012416BEO
SW00NAB 278 Next Avl Byte	001A
SW00CFAS 268 Current F	00000000CC006000
SW00WKA 324 Space work area	1799EC00

Dump Number: 00000004      Dump File Path: /tpfdbgdumpB/009947.TI0001.B

CPU ID: B      Terminal: D40305  
CPU Complex Time: TI0001      Program: DMPV  
Sub-System: BSS      Object: DMPV  
Sub-System User: HPN      Displacement: 061A  
I-Stream: 1      Program Status: 0715200180000000 00000000947D161A

DF Insight

# Dump Viewer

## View Dump Source

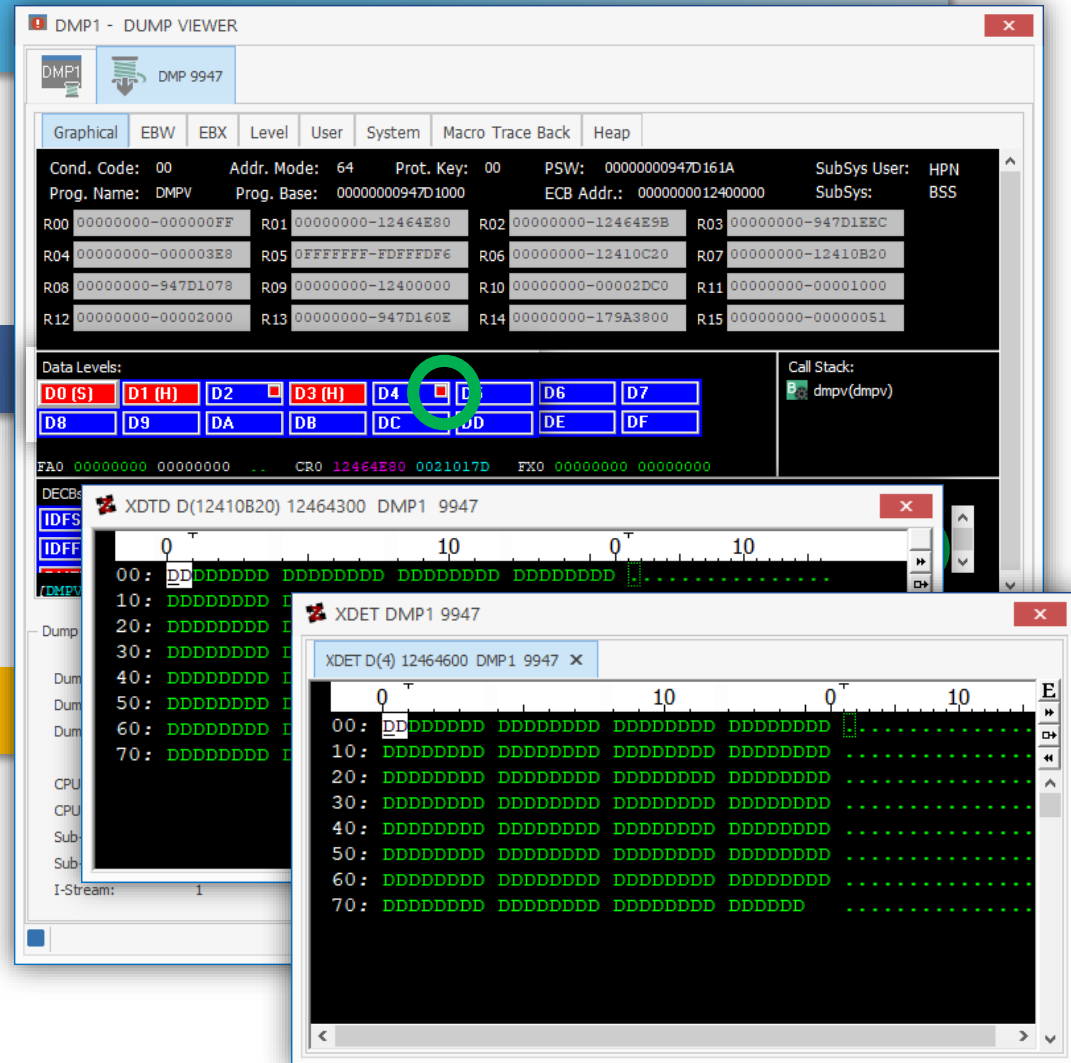
- ▶ Option to open read-only source file, if available, showing execution line where dump occurred.

## View TPDFDF in DF Insight

- ▶ Click on DF Insight in the Dump Viewer to view TPDFDF just as you would when tracing an ECB.

## View Detached Core Blocks

- ▶ Double-click a detached core block in a data level or DECB to open and view it in a Core Block window.



## What is it?

- ▶ Updates the collected data from the z/TPF machine in the customer DB in the customer defined structure through Jenkins (previously a manual step was required via a Web UI to select options & upload data to DB)

## Benefits

- ▶ No manual intervention required - faster, with no manual errors

## How it works

- ▶ Database table is customizable based on the customer need
- ▶ Data update event triggering is customizable. Can be based on...
  - ▶ **The new data from z/QDC**
  - ▶ **Time initiated**
  - ▶ **Manual trigger**



## What is it?

- ▶ zTPFGI now supports user customization of editor colors, along with an option for an overall dark color scheme

## Benefits

- ▶ Greater user productivity
- ▶ Greater accommodation of different visual abilities
- ▶ Greater user satisfaction

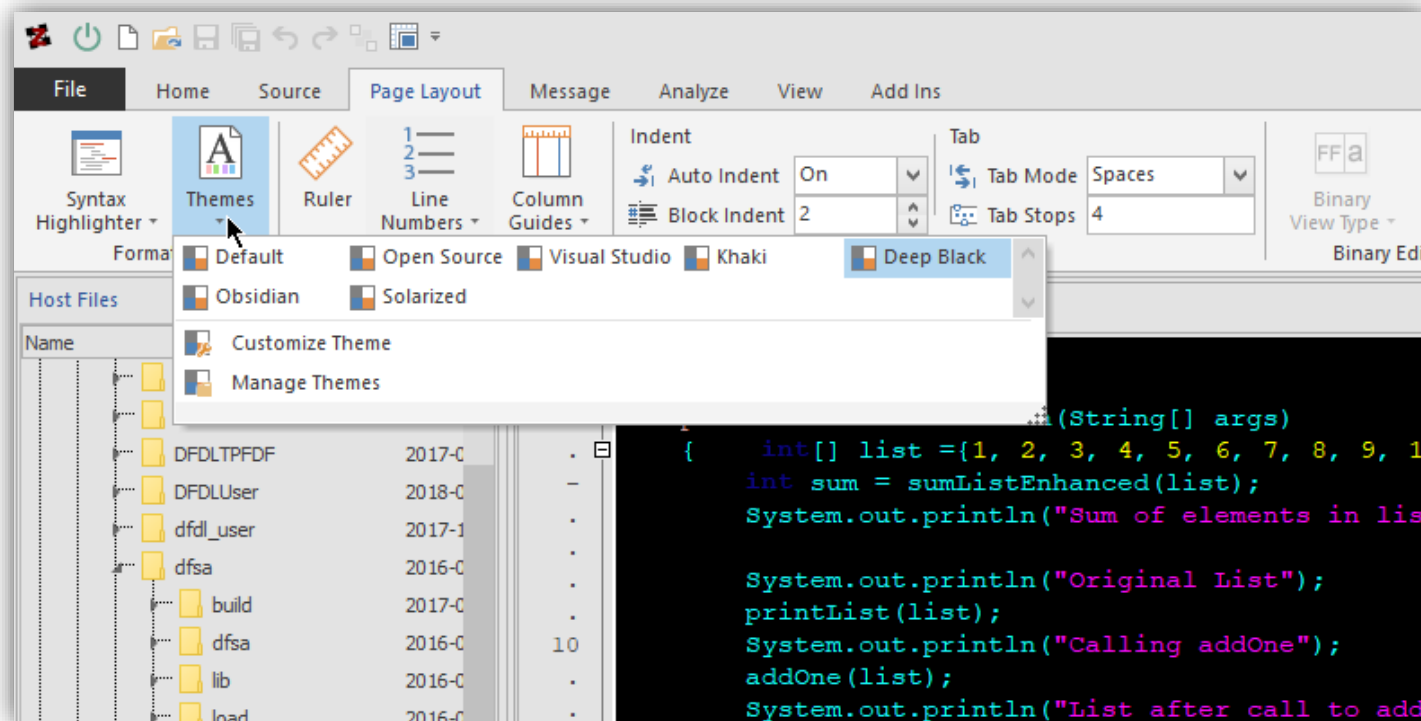
## How it works

- ▶ A greater range of standard editor themes now offers an array of non-white editor backgrounds
- ▶ In addition, editor themes may be duplicated, customized, and even shared
- ▶ An additional overall color scheme offers a darker choice for the ribbon and other menus and windows

# Color Enhancements – Standard Editor Themes

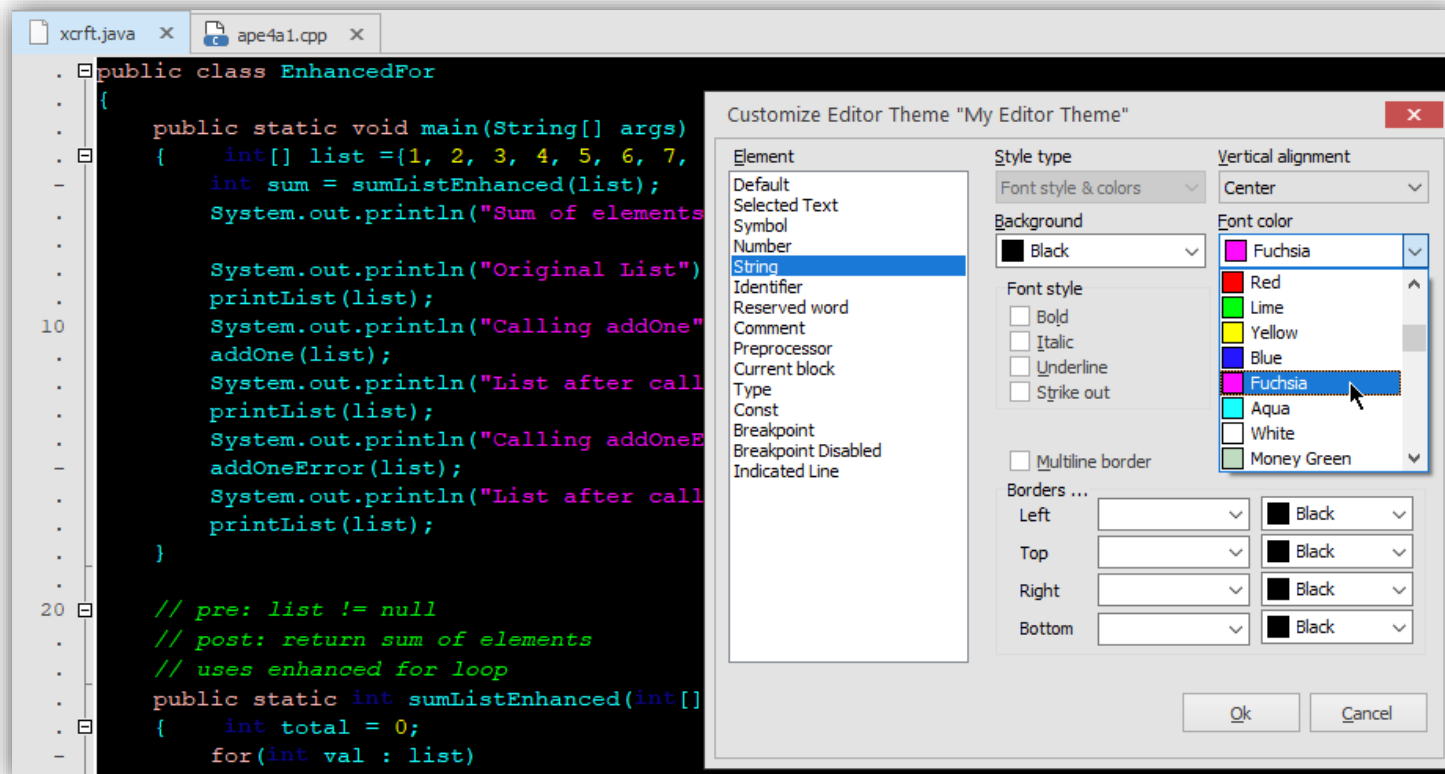
Color

- ▶ Users can hover over an expanded gallery of standard editor themes to instantly preview different syntax highlighting choices
- ▶ A single click makes the choice for that file type
- ▶ A preference is also available to change the default editor theme, affecting every file type without a different choice



# Color Enhancements – Custom Editor Themes

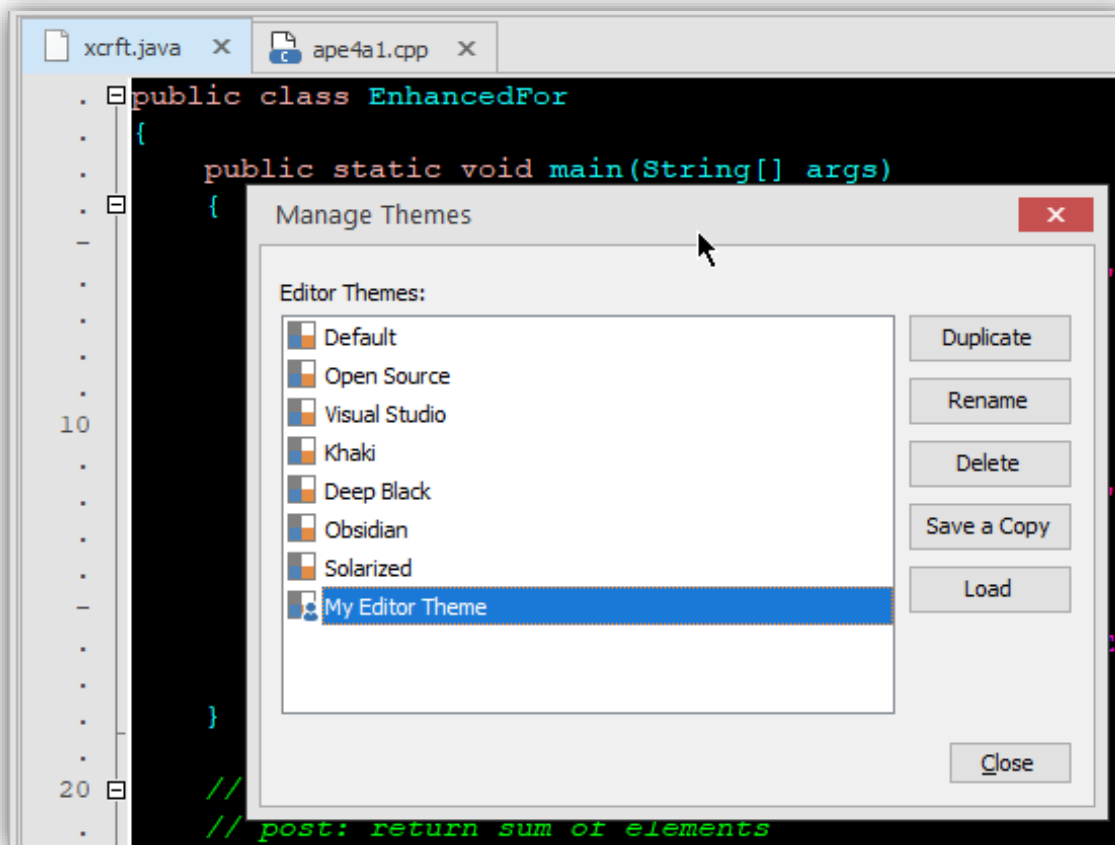
- ▶ Users can customize duplicate standard editor themes and customize every syntax element
- ▶ Even breakpoint and other colors can be customized
- ▶ Color and style choices are previewed instantly in the editor



# Color Enhancements – Editor Theme Management

Color

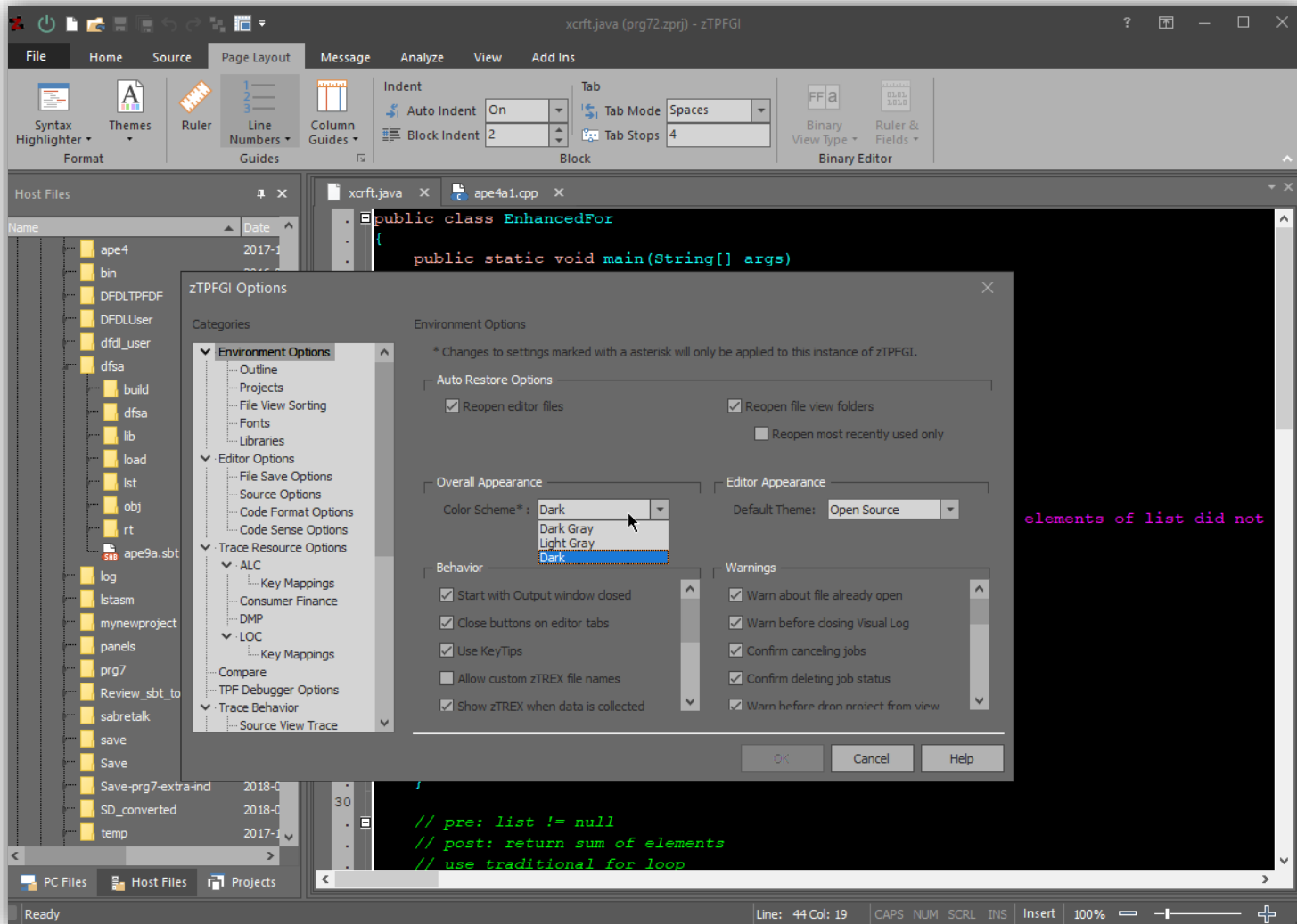
- ▶ Users can manage their custom editor themes
- ▶ They can also save a copy for co-workers, who can load it into their zTPFGI UI



# Color Enhancements – Dark Color Scheme

Color

- ▶ The new Dark color scheme affects the non-editor areas of zTPFGI





## ▶ ALC

- ▶ Message Capture: facility to Capture and replay ALC inputs
- ▶ Create ECB - Support to attach input message block to D0
- ▶ Execute large entries

## ▶ DF Insight

- ▶ Highlight current LREC in overlay view
- ▶ Display complete DF file including all forward chains

## ▶ Dump Viewer

- ▶ Support multiple sort options - Sort by date/time, program name

## ▶ Trace

- ▶ Async Trace support for C functions
- ▶ Performance improvements in Async
- ▶ Modify debugger to catch overwrites of system area of ECB
- ▶ Shortcut key for ECB

## ▶ Usability

- ▶ Option to open a file in desired mode (text, binary)\*
- ▶ Font size support for Host Files, PC Files, Projects, & Other Views\*
- ▶ Tooltips for view tabs\*
- ▶ Personalize system names for private labs\*
- ▶ Ability to copy paste variable visible in variable/watch list for C program\*



Questions?

TSI



MOVE FORWARD · TOGETHER

Thank you