Task Flow Recorder
for CICS

z/OS Installation Guide

For CICS TS Releases 4.2 – 5.4

Tel. ++1-734-846-0549
++972-52-664-1157

info@cicsrecorder.com

Note

TFR distribution library supports CICS TS releases 4.2 through 5.4.
Older CICS releases are supported as well, but require a different distribution library.
The Bundle that provides user interface via WebSphere Liberty Server requires CICS TS release 5.1 and up. If you are using CICS TS 4.2, you will need to install our .Net GUI in your IIS server.

Extract the Distribution File

Extract the file TfrInstallation.zip into a folder of your choice. A sub-folder named `TfrInstallation` is created. Examine its content.

Transfer TFR Libraries to z/OS

Transfer the three .XMIT files to your z/OS system (binary, record size=80, fixed).

```
TFR.LOAD.XMIT
TFR.LOAD5.XMIT
TFR.USERSRC.XMIT
```

Enter the TSO command:

```
RECEIVE INDSN('TFR.LOAD.XMIT')
```

When asked, please reply:

```
DSNAME('xxxx.TFR.LOAD')
```

where *xxxx* is a high-level qualifier of your choice.

Receive the other library (`TFR.USERSRC.XMIT`) as well.

Note: `TFR.LOAD` is a PDS library that contains all the load modules of the product: Assembler, COBOL and BMS mapsets. The COBOL programs in this library were compiled using the COBOL 4.2 compiler.
`TFR.LOAD5` is a PDSE library that contains all COBOL modules of TFR compiled using the COBOL 5.2 compiler. If you prefer to use this version of the programs, please concatenate `TFR.LOAD5` first. Otherwise, you do not need this library.

Copy the CICS Bundle to the UNIX file System (for CICS TS 5.1 – 5.4)

Note: if you are using CICS TS 4.2, you cannot use our Liberty Server GUI. In such a case, please skip this chapter. Instead, please install TFR .Net GUI in IIS, as described in the book **TaskFlowRecorder_InstallationDotNet_AlgoriNet.pdf**.

The bundle project was prepared for CICS releases 5.2 and above.
If you are using CICS TS 5.1, please make the following adjustment:

- Extract **com.cicsrecorder.tfr.bundle.zip** in your PC.
- Edit file **com.cicsrecorder.tfr.warbundle** using Notepad or another ASCII editor.
- Change **jvmserver="DFHWLP"** to **jvmserver="DFH$WLP"** and save.
- Right-click on the folder com.cicsrecorder.tfr.bundle.
  Send to | Compressed (zipped) folder.
  A new zip file is created: com.cicsrecorder.tfr.bundle.zip.
  Use it instead of the original.

From TSO option 6, invoke the z/OS Shell using command **OMVS**.
Create a directory in z/OS UNIX, for example:
mkdir /u/tfr

Using FTP from a command prompt window or another tool, transfer the file com.cicsrecorder.tfr.bundle.zip into the new directory:
ftp 112.113.114.115 (your z/OS IP or name)
(enter userid and password)
bin
cd /u/tfr
put com.cicsrecorder.tfr.bundle.zip

Back in the z/OS Shell window:
cd /u/tfr
jar xf com.cicsrecorder.tfr.bundle.zip

A sub-directory is created: **/u/tfr/com.cicsrecorder.tfr.bundle**.
It contains the bundle files and a WAR file.
At this point, you may delete the zip file:
rm com.cicsrecorder.tfr.bundle.zip

Add TFR Load Library to RPL

Add library ***xxxx*.TFR.LOAD** to **//DFHRPL DD** in every CICS region you wish to use the product.

Optionally: Add PDSE library xxxx.TFR.LOAD5 before TFR.LOAD, if you prefer to use the COBOL 5.2 version of TFR's COBOL modules.

Examine USERSRC Library

The USERSRC library contains JCL job streams, sample JCL procedures, sample programs, and copybooks. For your convenience, the members of USERSRC library can also be found in ASCII format in the folder USERSRC of the installation folder. You may open them using Notepad or other ASCII editors.

Choose a Three-Letter Prefix If 'TFR' is already in Use

The names of all the product's programs, transactions, file and DB2 entries begin with the three letters 'TFR'.
Please check whether you already have other transactions or programs which begin with the same three letters:

```
CEMT I TRAN(TFR*)
CEMT I PROG(TFR0*)
```

If you found nothing, you are fine; please skip the rest of this paragraph.

If you already use the prefix TFR, it is necessary to rename the product's resource names to another prefix. In such a case, please do the following:

- Choose an alternative three-letter prefix. In our example, you choose *ABC*.
- Edit job JRENAME.
- **CHANGE ALL XYZ *ABC*** (where *ABC* is the 3-letter prefix of your choice).
- Submit the job.
- **CHANGE ALL TFR *ABC*** in all JCL streams and JCL procedures in USERSRC library (besides JRENAME).
- Rename the WAR file and the bundle files similarly.

Define the VSAM File

- Edit job JDEFVSAM.
- Set the VOLUME parameter.
- Adjust the high level qualifier of DSNAME to your site naming conventions.
- Submit.

Define CSD entries (programs, transactions, mapsets, file, DB2entries)

- Edit job **JCSD**.
- Adjust the DSN of STEPLIB and DFHCSD before submitting.
  The VSAM file TFRFILE should be defined local in one region and REMOTE in all other regions.
  In the REMOTESYSTEM parameter, please specify the SYSID of the region where the file TFRFILE is going to be local.
  We recommend using the same region where Liberty Server is running.
- In the DEFINE BUNDLE command, make sure the attribute BUNDLEDIR points to the UNIX directory where TFR bundle had been saved earlier.
- If you are using CICS TS release 4.2, you cannot use the GUI under Liberty Server. In such a case, do not define the URIMAP and the BUNDLE resources.
  Instead you have to DEFINE TCPIPSERVICE, and use TFR .Net GUI. Please consult our technical support.
- Submit. Upon job completion, GROUP(TFR) is created in the CSD.
- **CEDA INSTALL GROUP(TFR)** in all CICS regions.
- Add the group to an active LIST. For example:
  CEDA ADD GROUP(TFR) LIST(XYZLIST)
- Assert that the programs are accessible (if the library is already in the RPL).
  CEMT S PROG(TFR*) NEW
- Assert that the product's VSAM file is open and enabled.
  CEMT I FILE(TFR*)

Bind a DB2 Program – If You Use DB2

If your CICS applications issue DB2/SQL calls, TFR retrieves their SQL statements from your DB2 catalogue. This is done by one of TFR modules, program TFR00CP. In order for it to run at your site, please bind that program.

- The DBRM member is ***XXXX*.TFR.USERSRC(TFR00CP)**.
- Edit job JBIND0CP to BIND program TFR00CP or use your own JCL for binding.
- Specify the correct library name in the LIBRARY parameter, and your TSO userid in the PROFILE parameter.
- Submit.

Complete the Installation

- Sign in to CICS in a 3270-terminal.
- Run transaction **TFR**. You should see the "Customization" page. Your userid is listed as the first Administrator.
- Specify the userid of additional product administrators if you wish.
- In the bottom line, enter the 4-character **Site License** provided by the vendor.
- Set **Hostcode** to the value specified in your 3270 emulator under menu option:
  Communication | Config | Session parameters.
  The default value '037 ' is for English-USA.
  Hostcode determines the transformation of non-English symbols between EBCDIC and Unicode, when TFR displays values on the browser.
- Enter command 'S' to save the updated profile, and press <Enter>. You should get a message that the customization profile has been saved.
- If you entered a wrong license code, you will get a message.
- Press PF3. You should see TFR home page.

Invoke TFR from your browser

- Make sure Liberty Server is defined and enabled. Check using:
  CEMT I JVM(DFHWLP)
- If not defined, please install its group:
  CEDA INS GROUP(DFH$WLP)
  You must customize the JVMProfile and server.xml, and then enable the JVM.
- Assert Liberty Server is accessible.
  Point your browser to the IP address or name of your z/OS system, and the port number of Liberty Server (9080 by default):
  http://112.113.114.115:9080
  You should see Liberty welcome page. If not, check the URL and the JVM status.
- Point your browser to TFR application home page under Liberty:
  http://112.113.114.115:9080/com.cicsrecorder.tfr
  You should see TFR Login page. If you do, installation process is complete.
  If you don't, please check the attributes of BUNDLE(TFR).
- Enter your CICS user id and password. You will get the list of recording sessions that you had run (an empty list if this is your first time).
- Click the "New Session" tab.
  On the right, you can see a list of all CICS regions where TFR had been installed, and are connected to the current region via MRO. Please examine the list and make sure all the intended regions are listed.
- Fill the form, Submit your first recording session.
- Follow the guidance in "TFR Walkthrough" document, and enjoy the product.

Preparing JCL Procedures for Advanced Features

At this point, Task Flow Recorder is operating well. You may take a pause and enjoy the product.

The following capabilities of TFR require running preceding batch jobs:

Feature	Required action
Display file records, commarea, TS Queue items and containers in field-by-field format (instead of plain bytes), using the COBOL structures (copybooks) which describe them.	Run a TFR utility that analyzes each copybook, and uploads its mapping to TFR data dictionary.
Trace and display the values of selected program variables or the complete working-storage in every stage throughout the program's execution.	Run a TFR utility that analyzes the program's compile listing and uploads its storage mapping to TFR data dictionary.
Execution flow of your program will contain not just CICS and DB2 calls, but also every visit in every COBOL paragraph and section.	Recompile your program, and add a TFR pre-compiler step.

For these purposes, TFR provides sample JCL procedures, which combine running of TFR utilities and the COBOL compiler in specific order and specific parameters.

Most data centers create their own JCL compile procedures, and use their own COB.SYSLIB and LKED.SYSLIB libraries. Therefore, we advise you ***not*** to run the provided procedures. Instead use them as a reference that will help you modify your own compile procedures, with the additional steps shown.

Following is an explanation about each of the above-mentioned goals, and how to use the sample JCL procedures.

Map various records using their associated COBOL structures (copybooks)

When TFR records the execution flow of a task, it can also record and display the content of file records that had been read or written, TS Queue items, commarea, containers or parameters that have been passed between tasks. In most cases, the layout of these records is described by a COBOL structure which is defined in a separate PDS member.
If you tell TFR which copybook is maps a particular file record or commarea, TFR can display that data field by field, formatted by the associated structure.

VSAM-Handy uses record mappings for practically everything it does.

TFR and VSAM-Handy are unable to access your SYSLIB libraries directly from a CICS transaction. Therefore, it is necessary to introduce each of these copybooks to TFR up front, via a batch job.

The task of uploading and analyzing a COBOL structure into TFR's Record Mapping Archive, is performed by the utility **TFRMAPCB**. This utility program does not read the COBOL structure directly. Its input is the compile listing file (i.e. `//COB.SYSPRINT`) of a tiny program that contains nothing but a working-storage section with this copybook.

This is precisely the task performed by the sample JCL procedure **TFRPMAP**. The procedure compiles a COBOL program that is generated by concatenating members `TFRMAPC0, TFRMAPC1, TFRMAPC2, TFRMAPC3` and the copybook itself in `//COB.IN DD`.
Please note that neither the CICS Translator nor the Linkage Editor is invoked. No load module is being created by the procedure. Its purpose is merely to produce a compile listing and pass it on to program `TFRMAPCB`.
The program receives a parameter that contains several attributes:

```
PARM='M,&MAPPING,&FILE,&PREFIX,&VALIDPGM'
```

Please refer to *TFR Walkthrough Guide* for full explanation about each attribute.

Prepare COBOL Programs for Variable Tracing or Working-Storage Snapshots

When a recording session is submitted, users can ask TFR to trace the values of specific variables in the working-storage and linkage sections of a program, or take complete working-storage snapshots after each CICS or DB2 command.
This capability is only available for programs whose storage mapping is known to TFR.

The task of uploading a program's storage mapping into TFR's Program Storage Archive, is performed by the utility **TFRMAPCB**. The input that utility TFRMAPCB requires is your program's compile listing (i.e. its `//COB.SYSPRINT DD`).

If your company already maintains a PDS library with the latest compile listing of each program, TFRMAPCB can read the listing from there. In such a case, there is no need to compile your program. Simply use member **TFR.USERSRC(JLISTING)** as a sample. Adjust the program name and the listing library name.
TFRMAPCB expects the listing PDS to be either `LRECL=133,RECFM=FB` or `LRECL=137,RECFM=VB`.

If you do not have a listing library, you have to add an `EXEC PGM=TFRMAPCB` step to your COBOL/CICS compile procedure.

Please use member **TFR.USERSRC(TFRPLIST)** as an example for a typical COBOL4.2/CICS compile procedure which was altered to upload the program's storage map to TFR's Program Storage Archive. The archive is the same VSAM file that was created in the beginning of the installation process, using job JDEFVSAM.

Here are the changes to be made:

- In the COBOL compile step you should have:
  ```
  //COB      EXEC PGM=IGYCRCTL
  . . .
  //SYSPRINT DD SYSOUT=*
  ```
  Change `SYSPRINT` so that the listing is passed on:
  `//SYSPRINT DD DSN=&&PRINT,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(3,1))`

- Since the user wants to see the listing, copy it to `SYSOUT=*`. Add the step:
  ```
  //COPY     EXEC PGM=IEBGENER
  //SYSUT1   DD   DISP=(OLD,PASS),DSN=&&PRINT
  //SYSUT2   DD   SYSOUT=*
  //SYSIN    DD   DUMMY
  //SYSPRINT DD   DUMMY SYSOUT=*
  ```

- At the end of the job, please add the following step:

```
//MAPPING  EXEC PGM=TFRMAPCB,COND=(4,LT,COB),PARM='K,&MEM'
//STEPLIB  DD   DISP=SHR,DSN=SYSADM.SAFE296.TESTLIB
//INP      DD   DISP=(OLD,DELETE),DSN=&&PRINT
//ARCHIVE  DD   DISP=SHR,DSN=HLQ.TFR.TRACE
//SYSPRINT DD   SYSOUT=*
```

  The parameter `&MEM` is the member being compiled. Your procedure might be using a different name, so adjust accordingly.

  After running TFRMAPCB, your program is listed in the "Programs" tab in TFR dashboard. When you invoke the "New Session" tab, that program appears in the dropdown list of available programs. That means TFR can trace its variables.

Record each visit in every COBOL paragraph and section

During a task recording session, TFR records every CICS command or database call. It can also detect a dynamic call, static call or a "GOBACK" command. A PERFORM statement into a COBOL paragraph or section is not recorded, since TFR's CICS Global User Exits do not receive control at that point in time.
Nevertheless, TFR provides a pre-processor program (TFRPREPR), which inserts into the application's code a dummy CICS call after each label, and passes the label name as a parameter. This way, TFR can record the visit to the paragraph, and it will later show up in the Execution Flow page of the task.

In order to achieve this goal, you should run TFR pre-processor program **TFRPREPR**, and then continue with the regular compile steps (CICS translator, compiler and LKED).

JCL procedure **TFR.USERSRC(TFRPLABL)** is an example. Use it as a reference for the changes to be made in your own COBOL/CICS compile procedure.

Your procedure is likely to begin with `EXEC PGM=ISRLEMX` that reads your source member and builds a file that is passed to the CICS translator, `EXEC PGM=DFHECP1$`.

```
//LEMX   EXEC PGM=ISRLEMX,
//       PARM=('COB,&MEM,B,N,E,4, ,00,ENG,4,7,1,/,SYSDA')
. . .
//ISRLEXPD DD DISP=(,PASS),DSN=&&LEMX,UNIT=SYSDA,
//       SPACE=(TRK,(10,10)),DCB=BLKSIZE=120
//TRN    EXEC PGM=DFHECP1$,REGION=3M,PARM='&SP'
//SYSIN  DD DISP=(OLD,DELETE,DELETE),DSN=&&LEMX
```

Insert an `EXEC PGM=TFRPREPR` step between these two steps, so that the output of `ISRLEMX` is passed to `TFRPREPR`, whose output is then passed to `DFHECP1$`:

```
//LEMX     EXEC PGM=ISRLEMX,
//      PARM=('COB,&MEM,B,N,E,4, ,00,ENG,4,7,1,/,SYSDA')
. . .
//ISRLEXPD DD DISP=(,PASS),DSN=&&LEMX,UNIT=SYSDA,
//         SPACE=(TRK,(10,10)),DCB=BLKSIZE=120
//PREPROC EXEC PGM=TFRPREPR
//STEPLIB DD DISP=SHR,DSN=HLQ.TFR.LOAD
//INP     DD DSN=&&LEMX,DISP=(OLD,DELETE)
//OUT     DD DISP=(,PASS),DSN=&&PREP,UNIT=SYSDA,
```

```
//         SPACE=(TRK,(10,10))
//TRN      EXEC PGM=DFHECP1$,REGION=3M,PARM='&SP'
//SYSIN    DD DISP=(OLD,DELETE),DSN=&&PREP
//SYSPUNCH DD DSN=&&SYSCIN,DISP=(,PASS),UNIT=SYSDA,
//          DCB=BLKSIZE=400,SPACE=(CYL,(50,50))
. . .
```

The rest of the procedure remains as it was.

If you want TFR to record COBOL labels and also trace program variables, then you should add the step `EXEC PGM=TFRMAPCB` as explained earlier, and pass it the compiler's listing `COB.SYSPRINT` as input.

Browse the program code while examining its execution flow

When the task execution flow screen is displayed, TFR can display the compile listing for each COBOL, Assembler or C program which was invoked throughout the task. This capability works if your data center saves the compiler's **//SYSPRINT DD** in a PDS library. If you tell TFR the name of that library (or libraries), TFR looks for a member with the same name as the program. If the compile listing member is found, the program name column for each event is shown in bold letters, and is clickable. When clicked, TFR opens a new window, and displays the procedure division portion of the listing. The command which was clicked is highlighted.

The names of your z/OS listing libraries should be listed in TFR properties file.

- Check WebSphere Liberty Server profile. The file is called DFHWLP and should be located in a directory called `JVMProfiles`. Find the value of `WORK_DIR`. This is the "current working directory" for TFR as well.
- In this `WORK_DIR` directory, please create an **ASCII** file called **`tfr.properties`**. If you are using TSO 3.17 (Udlist), use command EA (Edit Ascii); otherwise, the file will be in EBCDIC, and TFR's Java classes won't be able to read it.
- Enter property **`CompileListing`** with your compile listings PDS's. This is the library that is specified in compile step **`//SYSPRINT DD`** :
  `CompileListing=myhlq.cobol.listing`
  or multiple libraries, with a comma between them:
  `CompileListing=hlq.cob.list1,hlq.cob.list1,hlq.cob.list3`
- In case you save the properties file in a wrong directory, TFR will display the following message upon its initialization, letting you know where tfr.properties file is expected:
  "**`TFR properties file was not found in Liberty WORK_DIR. Please create: /u/vendor/tmp/tfr.properties`**"

If your program is re-compiled, make sure the new listing is saved in the listing library. TFR compares the timestamp of the saved listing and the actual timestamp of the load module.

Currently, TFR interprets listings created by COBOL 6.1, 5.2 or 4.2, Assembler or C compilers. In other words, the first line of the listing member must begin with one of the following:

```
PP 5655-EC6 IBM Enterprise COBOL for z/OS  6.1.0
PP 5655-W32 IBM Enterprise COBOL for z/OS  5.2.0
PP 5655-S71 IBM Enterprise COBOL for z/OS  4.2.0
15694A01 V1.13 z/OS XL C
High Level Assembler (in column 43 of 2nd line)
```

Technical Notes

- TFR uses standard CICS Global User Exits. It contains absolutely no system hooks or other "tricky code".

- All TFR's programs and user exits are thread safe and fully re-entrant.

- The global user exits issue no ENQ at all. Whenever there is a need to access shared memory, our code serializes the process using Assembler instructions "Compare and Swap" or "Test and Swap".

- The overhead imposed on CICS performance is negligible. Mostly some Assembler instructions, with only two or three CICS calls per recorded task.

- TFR does not have to run all day (unless your organization chooses otherwise). It only works when recording sessions are submitted, and then stops. In idle times, TFR imposes no performance overhead at all.

- Multiple recording sessions can work simultaneously. Users should not be concerned and may not even be aware of other recording sessions taking place by colleagues at the same time. Moreover, the same task might be recorded on behalf of two sessions.