Redbooks
ibm.com/redbooks

IBM z/OS V2R2: Diagnostics

Keith Winnard

Paul Robert Hering

Alvaro Salla

z Systems

IBM

Redbooks

International Technical Support Organization

IBM z/OS V2R2: Diagnostics

January 2016

SG24-8306-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (January 2016)

This edition applies to Version 2, Release 2, of z/OS (product number 5650-ZOS).

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of 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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DS8000®	Redbooks®	z/OS®
IBM®	Redbooks (logo) ®	z13™
IBM z13™	RMF™	
OS/390®	z/Architecture®	

The following terms are trademarks of other companies:

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.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get personalized notifications of new content
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android

Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication helps you to become familiar with the technical changes that were introduced into the Diagnostics area with IBM z/OS® V2R2.

This book is one of a series of IBM Redbooks publications that take a modular approach to providing information about the updates that are included with z/OS V2R2. This approach has the following goals:

- Provide modular content
- Group the technical changes into a topic
- Provide a more streamlined way of finding relevant information that is based on the topic

We hope you find this approach useful, and we welcome your feedback.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Keith Winnard is the z/OS Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and is keen engage with customers to understand what they want from IBM Redbooks publications. Before joining the ITSO in 2014, Keith worked for many clients and Business Partners in the UK and Europe in various technical and account management roles. He is experienced with blending and integrating new technologies into the traditional landscape of mainframes to improve business advantage.

Paul Robert Hering is an IT Specialist at the ITS Technical Support Center, Mainz, Germany. He provides support to clients with z/OS and z/OS UNIX related questions and problems. He has participated in several ITSO residencies since 1988, writing about UNIX related topics. Before supporting IBM OS/390® and z/OS, Robert worked for many years with the IBM VM operating system and its variations (VM/370, VM/HPO, VM/XA, and VM/ESA).

Alvaro Salla is a Senior IT Consultant for the ITSO. He has more than 40 years teaching and developing educational material covering the z/OS mainframe platform and consulting services that focus on performance. Alvaro has co-authored IBM Redbooks publications about IBM DS8000®, DFSMS, WLM, SYSPLEX, ABCs, IBM RMF™, and disaster recovery. He started with IBM in 1969.

Thanks to the following people for their contributions to this project:

Wendy Wang (z/OS Service Aids Development, Poughkeepsie Center) for her consultations, technical advice, and guidance on the SDUMP Ctrace enhancements content of this book.

Nicholas Jones (Software Engineer, z/OS Service Aids, Poughkeepsie Center) for his consultations, technical advice, and guidance on the SLIP and LOGREC enhancements content of this book.

Bob Haimowitz (Development Support Team, Poughkeepsie Center) for setting up and maintaining the systems, and providing valuable advice, guidance, and assistance throughout the creation of this book.

Rich Conway (Development Support Team, Poughkeepsie Center) for setting up and maintaining the systems, and providing valuable advice, guidance, and assistance throughout the creation of this book.

Peter Bertolozzi (Systems Management Specialist, Poughkeepsie Center) for setting up and maintaining the environments within syslab for residents to work in.

John Gierloff (Operations, Poughkeepsie Center) Residency set up and support.

Don Brennan (Development Support Team, Poughkeepsie Center) for setting up and maintaining the systems hardware used in the creation of this book.

Ella Buslovich (Graphics specialist, Poughkeepsie Center) for providing guidance and specialist graphics for this book.

Ann Lund (ITSO Administration, Poughkeepsie Center) for administrative support to enable the residency.

Cheryl Gera (ITSO Administration, Poughkeepsie Center) for managing the business operations for this book.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks publications

- For other publication in this z/OS V2R2 series:

  http://www.redbooks.ibm.com/Redbooks.nsf/pages/zos?Open
- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

SDUMP using CTRACE

This chapter describes the changes to SDUMP CTRACE that were implemented in the z/OS SYSDUMP (SDUMP) component with z/OS V2R2.

This chapter includes the following topics:

1.1 SDUMP CTRACE Enhancements

z/OS components can collect data about problem-related data and various events that occur within the component by using the Component Trace (CTRACE) service. The component must set up its own trace and use the appropriate options that are available to generate the trace records that are most applicable to the situation.

Each z/OS component that uses the CTRACE service has options available that vary from component to component.

In z/OS V2R2, SDUMP is enhanced to use CTRACE services to enrich serviceability by providing diagnostic data about SDUMP.

1.2 SDUMP CTRACE set up

SDUMP CTRACE uses the default parmlib member CTIDMP00. You can set up your own member or override the default member. You also can to Start, Stop, or Modify the options by using the **`TRACE CT`** command. After the data is captured, IPCS is available to view the SDUMP trace data, as shown in Figure 1-1.

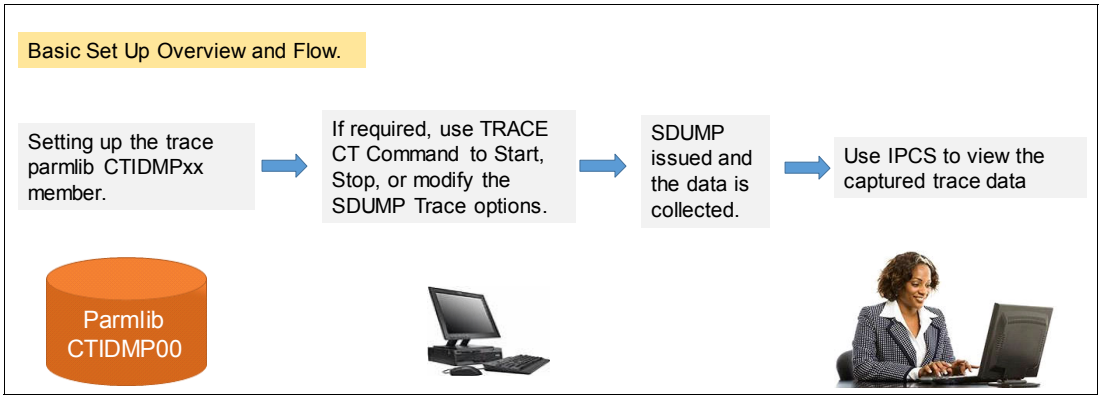

Figure 1-1 SDUMP CTRACE overview

1.2.1 CTIDMP00 parmlib member

A default CTIDMP00 member is supplied. Tracing is started by using the CTIDMP00 during the DUMPSRV address space initialization.

The following parameters are available to the SDUMP CTRACE parmlib member:

- ON or OFF: Determines whether the trace is on or off. The default is ON.
- BUFSIZE: Determines the size of the buffers for writing the trace records.
- OPTIONS: Determines the activity levels of tracing.
- WTR: Connects or disconnects the external writer for the trace.
- WTRSTART or WTRSTOP: Identifies the External Writer Started Task JCL and whether to open or close the data sets that are holding the trace data.

There are several considerations when you decide on your setup and configuration.

Collecting trace records

Trace records always are captured in a SVC dump if TRACE is set to ON. In addition, it can be written to data sets via an External Writer.

SDUMP CTRACE is captured at the end of the SDUMP capture phase and dumped at the end of the memory dump. If the trace data is needed at other times, you must request a new memory dump to collect the SDUMP CTRACE.

Buffer size

The buffer size determines the historical time spam of SDUMP events. When the buffer is full, CTRACE wraps the buffers and overwrites the oldest event records. Not all z/OS components can have the buffer size changed after the IPL; however, SDUMP is one of the components with which you can change the buffer size after the IPL. This change can be made by specifying another CTIDMPxx member or via the **`TRACE CT`** command. The CTRACE buffers include the following attributes:

- Location: 64-bit Common Storage Area
- Default size: 4 MB
- Range: 4 MB – 32 MB

OPTIONS parameter

The OPTIONS parameter can include the following possible values:

- `ALL`: Traces everything; that is, all the SDUMP events. This option is preferred because most SDUMP errors are difficult to reproduce. This option is the default.
- `MINIMUM`: Traces some SDUMP error events only.

> **Note:** Trace events are reverted to minimum when the trace is set Off (by a user or the CTIDMPxx member).

External writers

The following parameters apply to the External Writer options:

- `WTR(membername|DISCONNECT)`:
  - Connects or disconnects the component trace external writer and the trace.
  - The `membername` parameter identifies the member that contains the source JCL that starts the external writer.
  - `WTR(DISCONNECT)` disconnects the writer and the trace.
- `WTRSTART (membername)` identifies a member that contains source JCL for a started task, which the CTRACE uses to start the SDUMP component trace external writer and to open the data sets that the writer uses.

  > **Note:** The `membername` in the `WTR` parameter must match the `membername` in the `WTRSTART` parameter.

- `WTRSTOP(membername)` identifies the member that contains source JCL for a started task that the system used to start the SDUMP component trace external writer. The system also closes the data sets that the writer used.

Default CTIDMP00 member

The default CTIDMP00 member contains the parameters and values that are shown in Figure 1-2.

```
TRACEOPTS
ON
OPTIONS('ALL')
BUFSIZE(4M)
```

Figure 1-2 Default CTIDMP00 member

No CTIDMP00 member

If no CTDMP00 member is found, a default buffer size of 4 M is used and the trace options is set to MINIMUM.

1.3 TRACE CT command

The **TRACE CT** command can be used to display and modify options and to start and stop the SYSDUMP trace.

Displaying options

You can run the **DISPLAY TRACE** command before running a **TRACE CT** command to verify the supported options, as shown in Figure 1-3.

```
D TRACE,COMP=SYSDUMP
IEE843I 09.37.32  TRACE DISPLAY 738
        SYSTEM STATUS INFORMATION
 ST=(ON,0001M,00004M) AS=ON  BR=OFF EX=ON  MO=OFF MT=(ON,024K)
  COMPONENT     MODE BUFFER HEAD SUBS
 ------------------------------------------------------------
  SYSDUMP       ON   0004M
     ASIDS      *NOT SUPPORTED*
     JOBNAMES   *NOT SUPPORTED*
     OPTIONS    ALL
     WRITER     *NONE*
```

Figure 1-3 Display Trace command

You can override tracing options that are specified in the `CTIDMP00` parmlib member by another `CTIDMPxx` parmlib member that is identified on a **TRACE CT** command by using `PARM` parameter.

Example 1-1 shows the **TRACE CT** command in a `COMMNDxx` parmlib member. The command uses the PARMLIB member CTIDMP99, which might be set up to use different options than the default member CTIDMP00

Example 1-1 TRACE CT command in a COMMNDxx parmlib member

```
COM='TRACE CT,ON,COMP=SYSDUMP,PARM=CTIDMP99'
```

You can use the **TRACE CT** command to start, stop, or modify the SDUMP CTRACE. The syntax of the **TRACE CT** command is shown in Figure 1-4.

```
TRACE CT,ON,COMP=SYSDUMP[,PARM=mem]
         OFF,
         nnnnM,
TRACE CT,WTRSTART=membername[,WRAP|NOWRAP]
TRACE CT,WTRSTOP=membername[,FLUSH|NOFLUSH
```

Figure 1-4 Syntax of the TRACE CT command

In response to a **TRACE CT** command without the PARM parameter, the system prompts you to specify the component trace options that you want with a ITT006A message.

Example 1-2 shows more uses of the **TRACE CT** command.

Example 1-2 TRACE CT command examples

```
Change buffer size to 32M:
TRACE CT,32M,COMP=SYSDUMP

Change options using PARM=
TRACE CT,ON,COMP=SYSDUMP,PARM=CTIDMPxx

Change the OPTION to 'ALL'
TRACE CT,ON,COMP=SYSDUMP
*08 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
R 8,OPTIONS=(ALL),END

Turn off SDUMP CTRACE:
TRACE CT,OFF,COMP=SYSDUMP

Start Ctrace External Writer
TRACE CT,WTRSTART=WWWTR1
```

1.4 Viewing SDUMP CTRACE data

You can use the **IPCS** subcommand **CTRACE** to view the SDUMP component trace data. Four options are available: FULL, SHORT, SUMMARY, and TALLY

Samples of the SHORT and FULL options are described next.

SYSDUMP trace short version

To view the short version of the trace data, enter the following command:

```
IPCS CTRACE COMP(SYSDUMP) SHORT
```

The **IPCS** command produces the data report that is shown in Figure 1-5 on page 6.

```
COMPONENT TRACE SHORT FORMAT

COMP(SYSDUMP)

**** 01/22/2015

SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION

-------   --------  --------  ---------------  -------------

S7A0      WtDSVSST  0000007F  22:23:29.882181  Wait for DSVSSTECB to be post

S7A0      DmpStrtd  00000002  22:28:08.433737  Sdump started

S7A0      ClrStor   00000029  22:28:08.433753  Storage cleared for new dump

S7A0      EntyTSPR  00000013  22:28:08.433754  Entry at IEAVTSPR

S7A0      EntyTSPR  00000013  22:28:08.433759  Entry at IEAVTSPR

S7A0      SdumpPml  00000004  22:28:08.433760  Sdump PList after calling TSPR

S7A0      DumpSupp  00000006  22:28:08.433773  Dump is not being suppressed

S7A0      SchSD2TR  00000026  22:28:08.433774  Schedule TSD2 for SNAPTRC

S7A0      RTCTLUCT  0000002F  22:28:08.434757  Increment RTCTLUCT

S7A0      ScheTSD2  00000082  22:28:08.434757  Schedule SRB to TSD2
```

Figure 1-5 IPCS CTRACE Short format

SYSDUMP trace full version

To view the short version of the trace data, enter the following command:

```
IPCS CTRACE COMP(SYSDUMP) FULL
```

The **IPCS** command produces the data report that is shown in Figure 1-6 on page 7.

```
COMPONENT TRACE FULL FORMAT
COMP(SYSDUMP)
**** 01/22/2015
SYSNAME    MNEMONIC   ENTRY ID    TIME STAMP    DESCRIPTION
---------- ---------- ---------- ------------------ ------------------

S7A0       WtDSVSST   0000007F  22:23:29.882181   Wait for DSVSSTECB to be post

      ASID..... 0005            IssueMod. IEAVTSST  TCB...... 004DFD90
      RetnAddr. 89C0258A

S7A0       DmpStrtd   00000002  22:28:08.433737   Sdump started

      ASID..... 0029            IssueMod. IEAVTSDX TCB...... 004F8240
      RetnAddr. 8122A4BE

S7A0       ClrStor    00000029  22:28:08.433753   Storage cleared for new dump

      ASID..... 0029            IssueMod. IEAVTSDX TCB...... 004F8240
      RetnAddr. 8122A590
```

Figure 1-6 IPCS CTRACE Full format

1.4.1 Messages

The following messages were introduced to include support for the SYSDUMP CTRACE:

- `IEA053I` *`function_name`*`: Service` *`service_name`* `failed with RC:`*`return_code`* `RSN:`*`rsn_code`*

  A service was invoked that could not process the request. In the message text the function_name is the name of the function that invoked the service. The service_name is the name of the service that failed.

  A service was started that did not process the request. This message records this error. In the message, the text the name of the one is the name that started the service, and the name of the service is the service that failed.

- `IEA054I SDUMP CTRACE definition failed using CTIDMP00. RC=rc, RSN=rsn`

  The system cannot define the SDUMP component trace by using the CTIDMP00 parmlib member in the message text.

- `IEA055I Component trace option optname is not valid`

  The system encountered an incorrect option in the CTIDMPxx parmlib member or the TRACE CT command.

2

SLIP traps

This chapter describes the enhancements that were made in the Serviceability Level Indication Processing (SLIP) for z/OS V2R2 and includes the following topics:

The new enhancements might be of interest to operational staff who are responsible for automation because automation tracking can extend the effectiveness of SLIP trap processing.

2.1 CMD parameter enhancement

z/OS V2R2 **SLIP** enhancements include the **CMD** parameter on the **ACTION** parameter for a SLIP trap event. The option is available to all types of SLIP traps. You can include **ACTION=CMD** in a **SLIP SET** command, as shown in Figure 2-1.

```
SLIP SET,...,ACTION=CMD,CMD=('command1','command2',...),END
```

Figure 2-1 Using the ACTION=CMD command option

The **CMD** parameter is a required parameter when the **ACTION=CMD** option is specified. Use the format that is shown in Figure 2-2.

```
CMD='MVS COMMAND'
COMMAND='MVS COMMAND'
CMD=('MVS COMMAND')
CMD=('MVS COMMAND','MVS COMMAND')
CMD=('MVS COMMAND',('MVS ','COMMAND'))
```

Figure 2-2 Specifying the operations command

The **ACTION=CMD** option can be specified so that you can run a command. In Example 2-1, two display commands are to be issued when the slip trap's criteria are met.

Example 2-1 Issuing commands

```
SLIP SET,...,ACTION=CMD,CMD=('D CF','D GRS',),ML=1,END
```

Consider the following key points:

- **CMD** is allowed on all types of traps, including **PER** and error events.
- **CMD** is allowed with other actions, except **WAIT** and **IGNORE**.
- **CMD** is not allowed as a **REMOTE** option; however, **ROUTE** can be used as a command.
- Up to eight commands can be specified as such.
- Commands can be up to 126 characters long. Consoles might restrict the number of characters to less than 126. Therefore, you might have to use the **SET SLIP=xx** command to enter longer commands.
- Commands can be broken into parts with a second set of parenthesis.

SLIP message `IEE836D` has a new error description when more than eight commands are entered, as shown in Figure 2-3.

```
IEE836D SLIP {ID=xxxx|COMMAND} ERROR - keywd - err, COL nnn.
ENTER 1 KEYWORD, NULL LINE, OR 'CANCEL'.
```

Figure 2-3 Message IEE836D error description

You can respond manually to this error message or use automation to respond to the message and alert and report this error to the appropriate parties.

The user that is running the **SLIP** command must have authority to issue embedded commands. Consider the following points:

- The user that is entering an enabled **SLIP** command with **A=CMD** or the user that is enabling a disabled **SLIP** command that was created with **A=CMD** must have the authority to issue the entered command.
- The **SLIP** command issues an MGCRE macro and passes the **SLIP** issuer's authorization. The MGCRE macro allows a program to issue commands without operator intervention.
- Authorization is copied from the SLIP trap enabler.
- There is no guarantee in the order of completion of commands because command processing is asynchronous after a SLIP event. However, you can choose another option to issue a command that the automation subsystem recognizes and perhaps issue a REXX based script to control the command and message flow in the appropriate manner.

If multiple matches for a **SLIP** command with **A=CMD** are encountered, commands cannot be issued for every match but are issued at least once.

There is no default `MATCHLIM` value when the **A=CMD** is chosen.

z/OS V2R2 also introduces a Job Group function. Consider aligning this function with the **CMD** enhancements. Also, consider setting up a job group in a member. Multiple jobs perform various activities around the Sysplex that are based on the SLIP trap and issue a command (or multiple commands) to start an internal reader (or more) to submit job groups.

Migration and coexistence considerations

Consider the following points:

- **SLIP** commands that are routed to all consoles that contain **ACTION=CMD** fail syntax checking on pre-V2R2 systems.
- **SLIP** commands in `IEASLPxx` or `COMMNDxx` with **ACTION=CMD** fail syntax checking when processed on pre-V2R2 systems.

2.2 DESC optional parameter

The **SLIP SET DESC** optional parameter allows you to add a textual description to the dump title or wait state. The text can be 1 - 65 characters.

The **DESC** optional parameter is available for the following actions:

- WAIT
- SVCD
- SYNCSVCD
- STDUMP
- TRDUMP

It provides a better understanding of where and why these actions occurred.

The **DESC** keyword syntax is shown in the following example:

```
SLIP SET,...,ACTION=SVCD,DESC='dump description',END
```

Consider the following points regarding this example:

- **DESC** is optional
- **DESC** can be up to 65 characters in length

Example 2-2 shows setting a SLIP trap **WAIT** with a **DESC** keyword and forcing an abend X'0C1'.

Example 2-2 SET a SLIP trap WAIT

```
SLIP SET,C=0C1,ID=0001,A=WAIT,DESC='Catch 0C1',END

IEE844W SLIP TRAP 0001 MATCHED. ACTION=WAIT TYPE=RTM2
DESC: Catch 0C1
RTM2WA ADDR: 7F53CCF0 BEAR: 00000000_08288EA2
PSW: 07042000 80000000 00000000 08288ECE
CR 3-4: 00C00037 00000037
AR/GR 0: 00000000/00000000_01000000 1: 00000001/0A5CD588_01111111
2: 00000001/FFFFFFFF_01222222 3: 00000001/FFFFFFFF_01333333
4: 00000001/FFFFFFFF_01444444 5: 00000001/FFFFFFFF_01555555
6: 00000001/FFFFFFFF_01666666 7: 00000001/FFFFFFFF_01777777
8: 00000001/FFFFFFFF_01888888 9: 00000001/00000000_01999999
A: 00000001/00000000_0A5CD350 B: 00000001/00000000_0A5CD588
C: 00000001/00000000_0828AB78 D: 00000001/00000000_01DDDDDD
E: 00000001/00000000_01EEEEEE F: 00000001/FFFFFFFF_01FFFFFF
RESTART THE SYSTEM TO CONTINUE
```

Example 2-3 shows the use of the **CMD** and **DESC** keywords.

Example 2-3 Setting a SLIP trap and displaying it

```
· SLIP SET,C=0C1,ID=0001,A=(SVCD,CMD),CMD=('D SLIP','D GRS'),DESC='Command Action
and Dump',END

· D SLIP=0001
IEE735I 15.15.04 SLIP DISPLAY 523
ID=0001,NONPER,ENABLED
ACTION=SVCD,CMD,SET BY CONS CON3E0Y1,RBLEVEL=ERROR
MATCHLIM=1,0,COMP=0C1
CMD=('D SLIP', 'D GRS')
DESC=Command Action and Dump
```

Migration and coexistence considerations

Consider the following points:

- **SLIP** commands that contain the **DESC** keyword that are routed to all consoles fail syntax checking on pre-V2R2 systems.
- **SLIP** commands in `IEASLPxx` or `COMMNDxx` with the **DESC** keyword fail syntax checking when processed on pre-V2R2 systems.

2.3 For more information

For more information about SLIP traps, see the following resources:

- For information about operator commands to create, modify, or delete slip traps and information about the required parameters and general considerations, see *z/OS MVS System commands*, SA38-0666-03.

SLIP traps can dramatically affect system performance. Consult with technicians who are experienced and skilled in the use of SLIP to set effective SLIP traps and to use the minimum system resources to achieve your goal. These technicians can help to ensure that your SLIP traps are sufficiently accurate to capture first failure data capture or can set in motion the actions to increase the mean time between failures.

Always consider the use of the following parameters:

- `MATCHLIM` sets the maximum number of SLIP matches for this trap. When exceeded, the trap is disabled automatically. This parameter prevents unlimited recursive instances of the same SLIP trap processing-specific event.
- `PRCNTLIM` specifies a limit for **`PER`** processing by indicating the maximum percentage of CPU time that can be devoted to processing that is caused by **`PER`** interrupts.

- For an understanding of how to specify **`SLIP`** commands automatically at initial load time and other related options, avoid the restrictions that are found in other `PARMLIB` members (such as `COMMNDxx` and `IEACMDxx`) and use the `IEASLPxx PARMLIB` member as a single point of defining your SLIP traps. In addition, by adopting this policy, you can establish a more organized structure and process for your SLIP operational deployments. For more information, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380-05.
- For more information about how to use SLIP traps with other tools as part of a problem determination process, see *z/OS MVS Diagnostics:Tools and Service Aids*, GA32-0905-03.

You also can research and review the authority levels that are needed to use SLIP processing and related entities. There might be operational standards to which you must adhere within your organization. It is also advisable to be aware of the related security policies to ensure that you are operating within your organization's guidelines.

3

Log recording data set

Log recording data set (LOGREC) is a set of records (stored in a data set or in a log stream) that contain hardware and software error descriptions that relate to specific events.

z/OS V2R2 introduces updates for LOGREC.

This chapter includes the following topics:

3.1 LOGREC overview

When errors occur, the system records information about the error in the LOGREC data set or the LOGREC log stream. The records can describe hardware or software errors.

If you have a small single system, you can select a data set as the repository for the error records. However, if you have a sysplex, it is advantageous to reference a single source of error records that are appertaining to each system in the sysplex. In this case, you might prefer to chose one coupling facility log stream.

The following typical maintenance activities require an initial program load (IPL):

- Performing hardware, software, or operational maintenance on the DASD device on which the LOGREC data set is allocated.
- Switch from the `LOGSTREAM` option to the `DATASET` option after a System Logger outage, without having the LOGREC data set pre-allocated.
- Resize the `SYS1.Logrec` data set.
- Save an old `Sys1.Logrec` data set and switch to a new data set.

A more flexible solution is required to alleviate potential threats to availability. The following typical scenarios are the drivers for the enhancements that are introduced with z/OS V2R2:

- Enhance the **`SETLOGRC`** command to allow the following uses:
  - When switching to the `DATASET` option, allocate the data set. This action occurs even if your system is restarted with the `LOGSTREAM` or `IGNORE` options specified.
  - When switching from the `DATASET` option, deallocate the data set.
  - Allow specification of the `DATASET` or `LOGSTREAM` options resource.
- Enhance the `IEASYSxx LOGREC=` keyword to specify the log stream name.

The goal is to reduce the risk of maintenance activities by lowering availability levels and to provide a more flexible approach in the operational management of LOGREC. Ideally, new automation procedures might be introduced to use this flexibility to provide smoother operational running.

3.2 Changing the LOGREC specification

z/OS V2R2 introduces more functionality to the **`SETLOGRC`** command, as shown in Figure 3-1.

```
SETLOGRC {LOGSTREAM|DATASET|IGNORE|LOGSTREAM=lsname|DATASET=dsname}
```

Figure 3-1 SETLOGRC command syntax

The **`SETLOGRC`** command cannot be abbreviated. The `LOGSTREAM` option can be abbreviated to `LS`. In addition, the `DATASET` option can be abbreviated to `DS`.

You might want to issue this command to change the medium or to retain the same medium but use a different location by specifying a new name in the `DATASET=` or `LOGSTREAM=` parameters.

After the command is processed, one of the following results occur:

- If the change of medium is successful, the system issues an `IFB097I` message to indicate the change and the new medium to the requesting console.
- If the change is to the `DATASET` parameter:
  - If the change of medium is to `DATASET` and the system was not originally initialized with a data set specified, the system now dynamically allocates and opens the specified data set. If no data set name is provided, LOGREC defaults to the use of SYS1.LOGREC. If the system fails to open on the LOGREC data set, it issues the `IFB110I`, `IFB112I`, and `IFB099I` messages.
  - If the current medium is `DATASET` and `DATASET` is requested with a new data set name, LOGREC closes and deallocates the old data set. It then allocates and opens the new data set for LOGREC recording.
  - If the current medium is `LOGSTREAM` and `DATASET` is requested, LOGREC disconnects from the log stream. It then allocates and opens the data set for LOGREC recording. If problems arise when allocating the data set, LOGREC attempts to revert to log stream recording.
- If the change is to the `IGNORE` parameter:
  - If the current setting is `IGNORE` and `IGNORE` is requested, LOGREC issues the `IFB096I` message to the starting console to indicate that the wanted medium is the current setting.
  - If the wanted setting is to `IGNORE`, LOGREC error and environmental records are not recorded and are not provided in an ENF 36 signal.
  - If the current medium is `DATASET` and `IGNORE` is requested, LOGREC closes and deallocates the LOGREC data set.
  - If the current medium is `LOGSTREAM` and `IGNORE` is requested, LOGREC disconnects from the log stream.

  Suggestion: Use the `IGNORE` setting only in non-critical environments.

- If the change is to the `LOGSTREAM` parameter:
  - If the wanted setting is to `LOGSTREAM` and the connection to the log stream succeeds, the system issues the `IFB097I` message to indicate the successful change of medium from `LOGSTREAM` to `LOGSTREAM`.

    If the change of medium is unsuccessful, the system issues the `IFB099I` message. The system also issues the `IFB100E` message to indicate that the system logger is unavailable. LOGREC error and environmental records are buffered internally until the system logger becomes available. The buffer wraps around when the buffers are full.
  - If the connection is successful or when system logger and the log stream are available for writing, LOGREC writes records (including any buffered records) to the log stream.
  - If the current medium is `DATASET` and `LOGSTREAM` is requested, LOGREC closes and deallocates the LOGREC data set, then connects to the log stream.
  - If the current medium is `LOGSTREAM` and `LOGSTREAM` is requested, LOGREC disconnects from and then reconnects to the log stream. If a new log stream name is requested, LOGREC disconnects from the old log stream and connects to the new log stream.

Example 3-1 shows samples of the **SETLOGRC** command.

Example 3-1 Example of SETLOGRC commands

```
SETLOGRC LOGSTREAM=LOGREC.SYSPLEX1
SETLOGRC DATASET=MVSA.LOGREC
```

Example 3-2 shows the `IEASYSxx LOGREC=` parameter that is used at initial load to specify the option.

Example 3-2 Example of LOGREC parameter at IEASYSxx

```
LOGREC=LOGSTREAM=LOGREC.GROUP2
LOGREC=LS=SYS1.LOGREC.MVSX
```

The values for the `dsname` and `lsname` options can contain system symbols. When the `dsname` option is used, only one data set can be specified.

3.3 New messages

z/OS V2R2 introduces the following messages:

- `IFB110I` (as shown in Figure 3-2)

  Explanation: When you issued the **SETLOGRC** command to change the LOGREC recording medium to `DATASET`, the system encountered a problem.

IFB110I UNABLE TO function **LOGREC DATASET, NAME**: data-set-name, **RETCODE**: retcode **RSNCODE** rsncode

Figure 3-2 Message IFB11OI

- `IFB112I` (as shown in Figure 3-3)

  Explanation: This message contains dynamic allocation messages when the allocation of the LOGREC data set has problems after you use a **SETLOGRC** command.

IFB112I dynamic allocation messages

Figure 3-3 Message IFB112I

For more information, about the message fields, see *z/OS MVS System Messages, Vol 8 (IEF-IGD)*, SA38-0675-03.

4

DBX SIMD Support

The **dbx** utility is a source-level debugger for z/OS UNIX System Services. It provides an environment to debug and run C and C++ programs and to perform machine-level debug.

This chapter describes the new **dbx** functions and includes the following topics:

4.1 New support for dbx

z/OS V2R1 and z/OS V2R2 support the vector extension facility, single-instruction, multiple-data (SIMD), instructions that are available on IBM z13™ servers.

SIMD provides a powerful framework for developing new business analytics workloads, porting math-intensive workloads from other platforms, and accelerating business analytics workloads on IBM z13.

z/OS support includes enablement of vector registers on IBM z13, Mathematical Acceleration Subsystem (MASS), and Automatically Tuned Linear Algebra Software (ATLAS) support, and Language Environment enabled for C runtime functions.

In z/OS V2R2, **dbx** is enhanced to use SIMD capacity on UNIX System Services and provide the following functions:

- Display vector variables, registers, and instructions in debugging.
- Use vector variables and registers as a condition of breakpoints and trace.
- Compile code by using the following option to get SIMD support:

  `-qvector -qarch=11 -Wc,'FLOAT(IEEE)'`

4.2 Vector variables and registers display and change

This section provides an example of the use of the **dbx** utility to debug a sample program. Figure 4-1 shows the sample program in a **dbx** session.

```
 1 void main()
 2 {
 3   vector signed int int_1={1,2,3,4};
 4   vector signed int int_2;
 5   vector unsigned short short_1={11,12,13,14,15,16,17,18};
 6   vector bool long long long_1={22,23};
 7   vector unsigned char char_1={'a','b','c','d'};
 8   vector signed long long array_1[2]={{31,32},{33,34}};
 9   vector double z={0.1314, 897655.9};
10   vector unsigned char* pChar;
11   vector signed long long* pLong;
12
13   int c=2014;
14
15   c++;
16   int_2[3]=c;
17   pChar = &char_1;
18   pLong = &array_1[1];
19
20   return;
21 }
```

Figure 4-1 Sample program example

Example 4-1 shows the use of the **dbx** session for debugging the example program.

Example 4-1 Use of dbx for debugging the example program

```
$> c99 -go bin/example -qvector -qarch=11 -Wc,'FLOAT(IEEE)' example.c
$> dbx64
FDBX1111: dbx args: dbx64
FDBX1112: Current Directory: /u/hering
FDBX0089: dbx for z/OS with 64-bit support.
FDBX0399: Compiled: May 26 2015 at 02:43:13 (v2.2) (PROD)
FDBX0400: OS level: 02.00 02; LE level: 4.2.2; (Local)
...
FDBX0252: enter object file name (default is 'a.out', ^D to exit): bin/example
FDBX6432: Processing load module "bin/example"
...
FDBX6421: Loaded debug data from "/u/hering/example.dbg"
FDBX0150: Debug target is 31-bit.

(dbx64) stop in main
[1] stop in 'void main()'       File WTSC74:/u/hering/example.c, Line 3.
(dbx64) c
[1] stopped in main at line 3 in file "example.c"  ($t1)
    3     vector signed int int_1={1,2,3,4};
(dbx64) list
    4     vector signed int int_2;
    5     vector unsigned short short_1={11,12,13,14,15,16,17,18};
    6     vector bool long long long_1={22,23};
    7     vector unsigned char char_1={'a','b','c','d'};
    8     vector signed long long array_1[2]={{31,32},{33,34}};
    9     vector double z={0.1314, 897655.9};
   10     vector unsigned char* pChar;
   11     vector signed long long* pLong;
   12
   13     int c=2014;
(dbx64) st at 20
[2] stop at "example.c":20
(dbx64) c
[2] stopped in main at line 20 in file "example.c"  ($t1)
   20     return;
(dbx64) print int_1
(1, 2, 3, 4)
(dbx64) print long_1,char_1
(22, 23) "abcd"
(dbx64) print array_1
((31, 32), (33, 34))
(dbx64) whatis short_1
vector unsigned short short_1;
(dbx64) p *pChar
"abcd"
(dbx64) p *pLong
(33, 34)
(dbx64) unset $novregs
(dbx64) registers
$r0: 0x0000000014dea388   $r1: 0x0000000014dea344   $r2: 0x00000000890ef310
  $r3: 0x0000000000000002   $r4: 0x000000008930f23a   $r5: 0x00000000139dba48
  $r6: 0x00000000139dbbd4   $r7: 0x00000000139db098   $r8: 0x0000000000000030
  $r9: 0x0000000080000000  $r10: 0x00000000890ef242  $r11: 0x000000008930f148
 $r12: 0x0000000014de5188   $sp: 0x0000000014dea248  $r14: 0x0000000014dea248
 $r15: 0x00000000939db928
$psw0: 0x00000000078d2400 $psw1: 0x00000000939dba26   $pc: 0x00000000139dba26
```

```
   PSW: 078d2400 939dba26
        Instruction address: 0x139dba26
        Condition code: 2

FDBX0552: unset $noflregs to view floating point registers.
FDBX0553: unset $noflbregs to view IEEE floating point registers.
FDBX0557: unset $nofldregs to view decimal floating point registers.

  vreg   value in hex
   $vr0: 0x3fc0d1b7 1758e219 412b64ef cccccccd
   $vr1: 0x41120dd7 50429b6d 00000000 00000000
   $vr2: 0x40c90fda a22168c2 00000000 00000000
   $vr3: 0x4116a09e 667f3bcd 00000000 00000000
   $vr4: 0x40517cc1 b727220b 00000000 00000000
   $vr5: 0x40b504f3 33f9de65 00000000 00000000
   $vr6: 0x40a2f983 6e4e4415 00000000 00000000
   $vr7: 0x00000000 00000000 00000000 00000000
   $vr8: 0x00000000 00000000 00000000 00000000
   $vr9: 0x00000000 00000000 00000000 00000000
  $vr10: 0x00000000 00000000 00000000 00000000
  $vr11: 0x00000000 00000000 00000000 00000000
  $vr12: 0x00000000 00000000 00000000 00000000
  $vr13: 0x00000000 00000000 00000000 00000000
  $vr14: 0x00000000 00000000 00000000 00000000
  $vr15: 0x00000000 00000000 00000000 00000000
  $vr16: 0x00000000 00000000 00000000 00000000
  $vr17: 0x00000000 00000000 00000000 00000000
  $vr18: 0x00000000 00000000 00000000 00000000
  $vr19: 0x00000000 00000000 00000000 00000000
  $vr20: 0x00000000 00000000 00000000 00000000
  $vr21: 0x00000000 00000000 00000000 00000000
  $vr22: 0x00000000 00000000 00000000 00000000
  $vr23: 0x00000000 00000000 00000000 00000000
  $vr24: 0x00000000 00000000 00000000 00000000
  $vr25: 0x00000000 00000000 00000000 00000000
  $vr26: 0x00000000 00000000 00000000 00000000
  $vr27: 0x00000000 00000000 00000000 00000000
  $vr28: 0x00000000 00000000 00000000 00000000
  $vr29: 0x00000000 00000000 00000000 00000000
  $vr30: 0x00000000 00000000 00000000 00000000
  $vr31: 0x00000000 00000000 00000000 00000000

in main at line 20 in file "example.c"  ($t1)
0x139dba26 (main+0xfe)  4400c1ac     EX      0,428(,R12)
(dbx64) print $vr0
(1069601207,391701017,1093362927,-858993459)
(dbx64) print $vr0s
(16320,-11849,5976,-7655,16683,25839,-13108,-13107)
(dbx64) print $vr0c
('^?','{','J','^?','^?','^?','S','^?','^?','^?','^?','^?','^?','^?','^?','^?')
(dbx64) print $vr0c[6]
'S'
(dbx64) print $vr0c[1..2]
[1] = '{'
[2] = 'J'
(dbx64) assign int_2=int_1
(dbx64) print int_2
(1, 2, 3, 4)
(dbx64) assign z[0]=0
(dbx64) p z
```

```
(0.000000, 897655.900000)
(dbx64) assign short_1=int_1
(dbx64) p short_1
(0, 1, 0, 2, 0, 3, 0, 4)
(dbx64) assign long_1=short_1
(dbx64) set $hexints
(dbx64) p long_1
(0x100000002, 0x300000004)
(dbx64) unset $hexints
(dbx64) assign pLong=&long_1
(dbx64) p *pLong
(4294967298, 12884901892)
(dbx64) assign (*pChar)[0]='f'
(dbx64) p char_1
"fbcd"
(dbx64) assign array_1[0]=long_1
(dbx64) p array_1
((4294967298, 12884901892), (33, 34))
(dbx64) assign int_1={0,0,0,0}
assign int_1={0,0,0,0}
            ^ syntax error
(dbx64) assign int_1=0
FDBX0056: assign non-composite to composite
(dbx64) set $hexints
(dbx64) p $vr2
(0x40c90fda,0xa22168c2,0x00000000,0x00000000)
(dbx64) assign $vr2[0]=0x10
(dbx64) assign $vr2[1]=0x11
(dbx64) assign $vr2[2]=0x12
(dbx64) assign $vr2[3]=0x13
(dbx64) p $vr2
(0x00000010,0x00000011,0x00000012,0x00000013)
(dbx64) assign $vr1=$vr2
(dbx64) p $vr1
(0x00000010,0x00000011,0x00000012,0x00000013)
(dbx64) assign $vr0[0]=1024*2+1
(dbx64) p $vr0[0]
0x00000801
(dbx64) assign $vr31c[0]='a'
(dbx64) assign $vr31c[1]='g'
(dbx64) assign $vr31c[2]=0x88
(dbx64) p $vr31c
('a','g','h','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0')
(dbx64) assign $vr29s[1]=0xffff
(dbx64) p $vr29s
(0x0000,0xffff,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000)
(dbx64) assign $vr2x[0]=1
FDBX0614: "$vr2x" is not defined
(dbx64) c
FDBX0114: program exited
(dbx64) quit
$>
```

Table 4-1 lists the **dbx** subcommands that are used in Example 4-1 on page 21. For more information, see *z/OS UNIX System Services Command Reference*, SA23-2280-02.

Table 4-1 dbx subcommands

Subcommand	Description
assign	Assign a value to a variable.
cont	Continue program execution.
list	Display lines of the current source file.
print	Print the value of an expression.
quit	End the **dbx** debugging session.
registers	Display the value of registers.
set	Define a value for a **dbx** variable.
stop	Stop execution of a program.
unset	Delete a variable.
whatis	Display the type of program components.

4.3 Conditional stops and tracing

Figure 4-2 shows an example that demonstrates the use of the **dbx** utility to debug a sample program.

```
 1  void subFunc1(vector unsigned short input);
 2  vector signed int subFunc2();
 3
 4  void main()
 5  {
 6    vector signed int mainVar_1={1,1,1,1};
 7    vector unsigned short mainVar_2={2,2,2,2};
 8    vector unsigned long long mainVar_3[2]={{30,31},{56,57}};
 9
10    mainVar_2[1]++;
11    subFunc1(mainVar_2);
12    mainVar_1 = subFunc2();
13    mainVar_3[0][0]--;
14    mainVar_3[1][1] = 99;
15    return;
16  }
17
18  void subFunc1(vector unsigned short input)
19  {
20    input[7]=7;
21    return;
22  }
23
24  vector signed int subFunc2()
25  {
26    vector signed int rt={5,6,7,8};
27    return rt;
28  }
```

Figure 4-2 Sample program example2.c

Example 4-2 shows the use of the **dbx** session for debugging the second example program.

Example 4-2 Using dbx for debugging the example2 program

```
$> c99 -go bin/example2 -qphsinfo -qvector -qarch=11 -Wc,'FLOAT(IEEE)' \
> example2.c
CCN0000(I) Product(5650-ZOS) Phase(CCNEOPTP) Level(D150318.1512)
CCN0000(I) Product(5650-ZOS) Phase(CCNDRVR ) Level(D150318.1512)
CCN0000(I) Product(5650-ZOS) Phase(CCNEP   ) Level(D150318.1512)
CCN0000(I) Product(5650-ZOS) Phase(CCNETBY ) Level(D150318.1512)
CCN0000(I) Product(5650-ZOS) Phase(CCNECWI ) Level(D150318.1512)
CCN0000(I) Product(5650-ZOS) Phase(CCNEDWRT) Level(D150318.1512)
CCN0000(I) Library(elf     ) Phase(CCNEDWRT) Level(D150318.1512)
CCN0000(I) Library(dwarf   ) Phase(CCNEDWRT) Level(D150318.1512)
CCN0000(I) Library(ddpi    ) Phase(CCNEDWRT) Level(D150318.1512)
$> dbx64
...
FDBX0252: enter object file name (default is 'a.out', ^D to exit): bin/example2
FDBX6432: Processing load module "bin/example2"
...
```

```
FDBX6421: Loaded debug data from "/u/hering/example2.dbg"
FDBX0150: Debug target is 31-bit.
(dbx64) st at 11
[1] stop at "example2.c":11
(dbx64) trace if mainVar_1[0]>0
[2] trace if mainVar_1[0] > 0
(dbx64) c
FDBX0304: trace in example2.c ($t1):    7     vector unsigned short mainVar_2={2,2,2,2};
FDBX0304: trace in example2.c ($t1):    8     vector unsigned long long
mainVar_3[2]={{30,31},{56,57}};
FDBX0304: trace in example2.c ($t1):   10     mainVar_2[1]++;
FDBX0304: trace in example2.c ($t1):   11     subFunc1(mainVar_2);
[1] stopped in main at line 11 in file "example2.c"  ($t1)
   11     subFunc1(mainVar_2);
(dbx64) stop in subFunc1 if mainVar_2[3]==2
[3] stop if mainVar_2[3] = 2 in 'void subFunc1(vector unsigned short input)'
File WTSC74:/u/hering/example2.c, Line 20.
(dbx64) c
FDBX0304: trace in example2.c ($t1):   20     input[7]=7;
[3] stopped in subFunc1 at line 20 in file "example2.c"  ($t1)
   20     input[7]=7;
(dbx64) stopi if $vr0[1]<>100
[4] stopi if $vr0[1] <> 100
(dbx64) c
stopped in subFunc1 at 0x139db980 ($t1)
0x139db980 (subFunc1+0x58)  5810d0e0     L       R1,224(,R13)
(dbx64) p $vr0[1]
131074
(dbx64) delete all
(dbx64) stop if mainVar_3[0][0]!=30
[5] stop if mainVar_3[0][0] <> 30
(dbx64) c
stopped in main at line 14 in file "example2.c"  ($t1)
   14     mainVar_3[1][1] = 99;
(dbx64) tracei if mainVar_2[4]=0
FDBX0305: tracei ($t1): 0x139dbb6e (main+0x136)  4400c1ac     EX      0,428(,R12)
[6] tracei if mainVar_2[0x4] = 0x0
(dbx64) c
FDBX0305: tracei ($t1): 0x13a1ab5e (main+0x126)  4110d128     LA      R1,296(,R13)
FDBX0305: tracei ($t1): 0x13a1ab62 (main+0x12a)  e54c10000000 MVHI    0(R1),0
FDBX0305: tracei ($t1): 0x13a1ab68 (main+0x130)  e54c10040063 MVHI    4(R1),99
FDBX0305: tracei ($t1): 0x13a1ab6e (main+0x136)  4400c1ac     EX      0,428(,R12)
stopped in main at line 15 in file "example2.c"  ($t1)
   15     return;
(dbx64) c
FDBX0305: tracei ($t1): 0x13a1ab76 (main+0x13e)  4400c1b8     EX      0,440(,R12)
FDBX0114: program exited
(dbx64) quit
$>
```

Table 4-2 lists the **dbx** subcommands that are used in Example 4-2 on page 25 that are not listed in Table 4-1 on page 24. For more information, see *z/OS UNIX System Services Command Reference*, SA23-2280-02.

Table 4-2 dbx subcommands

Subcommand	Description
delete	Remove traces and stops.
stopi	Stop at a specified location.
trace	Print tracing information.
tracei	Print tracing information.

Related publications

The publications that are listed in this section are particularly suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks publications

The following IBM Redbooks publications provide more information that is related to the z/OS V2R2 updates. Note that some publications that are referenced in this list might be available in softcopy only:

- *z/OS V2R2: JES2, JES3, and SDSF,* SG24-8287
- *z/OS V2R2: Security,* SG24-8287
- *z/OS V2R2: Storage Management and Utilities,* SG24-8289
- *z/OS V2R2: Availability Management,* SG24-8290
- *z/OS V2R2: Performance,* SG24-8292
- *z/OS V2R2: Operations,* SG24-8305
- *z/OS V2R2: Diagnostics,* SG24-8306
- *z/OS V2R2: Sysplex,* SG24-8307
- *z/OS V2R2: UNIX System Services* SG24-8310
- *z/OS V2R2: User Interfaces,* SG24-8311
- *z/OS V2R2: ServerPac* SG24-8500

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at the following website:

ibm.com/redbooks

Other publications

The following publications are also relevant as further information sources:

- *z/OS V2R2 MVS Diagnosis: Tools and Service Aids*, GA32-0905
- *z/OS V2R2 MVS Diagnosis: Reference*, GA32-0904
- *z/OS V2R2 MVS Initialization and Tuning Reference*, SA23-1380
- *z/OS V2R2 MVS System Commands*, SA38-0666
- *z/OS V2R2 MVS System Messages*, Vol 6 (GOS-IEA), SA38-0673
- *z/OS V2R2 MVS System Messages*, Vol 8 (IEF-IGD), SA38-0675

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

(0.1”spine)
0.1”<->0.169”
53<->89 pages

SG24-8306-00

ISBN 0738441309

Printed in U.S.A.