Tuesday, October 27, 2026, 3:30 PM, Gardner Room

OpenVMS DEBUG Formatting Extensions

2026 OpenVMS Bootcamp
Portsmouth Sheraton
Portsmouth, New Hampshire USA

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Agenda –

- OpenVMS DEBUG EXAMINE on structures
- Good structure elements helpful in DEBUG, development, and production error reporting
- Write once; use often
- From Ruby: every structure/object has a formatter

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

References

- VSI OpenVMS System Management Manual
- VSI OpenVMS Debugger Manual
- VSI OpenVMS System Service Reference Manual: A–GETUAI
- VSI OpenVMS System Service Reference Manual: GETUTC–Z
- VSI OpenVMS DCL Dictionary: N–Z
- VSI OpenVMS RTL Library (LIB$) Manual

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Introduction

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Other Relevant Presentations

- Building Applications for Troubleshooting and Accountability
- Ex-PostFacto OpenVMS DEBUG
- OpenVMS DEBUG: A Powerful Tool for Sophisticated Debugging (2025 Bootcamp)

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

OpenVMS DEBUG Understands Structures – To a point

- C/C++ Enums – Yes
- Integers – displayed in present base
- String: <pointer>
- Fixed Length: ASCII (in quotes)
- Bit Masks: Numbers

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Good news; Bad News

- Better than one word at a time in hex/octal
- Does not automatically do char*
- Global numeric base, e.g., octal; hexadecimal; decimal
- Complex formats, e.g., UUID, MAC, IPv4 addresses
- Error/validity checks

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

OpenVMS DEBUG: Default Structure Display

```
DBG> examine @CurrentTransaction
*SAMPLE\formatTransaction\CurrentTransaction
    Signature:  421075225
    EntryCategory:      ET_TypeB
    Name:       16576
    Category:   'X435'
DBG>
```


© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Problems with Default OpenVMS DEBUG

- Pointers
- Radix
- Specials

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Varying Radices

- OpenVMS DEBUG allows setting display radix
- Choices are decimal, octal, hexadecimal
- What if no single choice is “just right”
- What about structures with masks, addresses, and special items, e.g., IPv4 addresss, UUIDs, MAC

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

UUID

- 128 bits (4 longwords)
- preferred format (hexadecimal digits):
  xxxxxxxx-xxxx-xxxx-xxxx-xxxxxx
- essentially useless as just a octaword

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

MAC

- 48 bits (3 words)
- preferred format (hexadecimal digits: xx-xx-xx-xx-xx-xx)
- essentially useless when displayed as words

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Network Addresses, e.g., IPv4

- 32 bits
- preferred display format (dotted decimal): d.d.d.d
- useless when displayed as a 32-bit longword
- Example: 3232235777 is 192.168.1.1

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Char *

- OpenVMS DEBUG displays the address in present radix
- Much more useful: 0xXXXXXXXX "string"
- Even more useful: 0xXXXXXXXX (d) "string"

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Character Descriptor

- OpenVMS DEBUG displays the descriptor contents
- Even more useful is pointer to descriptor;
  the descriptor contents (in appropriate radices;)
  and the described data
  Example: 0xXXXXXXXX -> 0xXXXXXXXX (typestring) (d) "st

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Answer: Extend OpenVMS DEBUG

- OpenVMS DEBUG CALL command allows invocation of user-written code from OpenVMS DEBUG commands

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Guidelines for OpenVMS DEBUG CALL Routine

- Presume minimal context
- DO NOT use language RTLs
- System services and LIB$ should be safe
- No writeable global data
- A CALL routine can be invoked at ANY time,
  e.g., process level; AST level; stack invalid

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

A Sample DEBUG CALL Routine

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

A Sample Program:

```
enum EntryTypeCategories {
        ET_TypeA,
        ET_TypeB,
        ET_TypeC,
        ET_MAXENUM};

typedef struct {
        unsigned int Signature;
        enum EntryTypeCategories EntryCategory;
        char *Name;
        char Category[4];
        } Transaction;
      . . .

int main()
{
        Transaction Transactions[100];
        Transaction *CurrentTransaction;


        CurrentTransaction = &Transactions[0];
        CurrentTransaction->Signature = Transaction_Signature;
        CurrentTransaction->EntryCategory = ET_TypeB;
        CurrentTransaction->Name = "XYZ1233";
        strncpy(&CurrentTransaction->Category[0], "X435",
            sizeof(CurrentTransaction->Category));

        formatTransaction(CurrentTransaction);
        formatTransaction(&Transactions[2]);
        BugCheck("Invalid State");
        return SS$_NORMAL;

        }
```


© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Output:

```
DBG> call formatTransaction(CurrentTransaction)
    Signature:  Transaction_Signature
    EntryCategory:      ET_TypeB
    Name:       (@16576; length 8) ”XYZ1233”
    Category:   'X435'
DBG>
```


© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Summary

- DEBUG Goal: Eliminate routine scratch paper
- Properly written formatting routines are multi-role – OpenVMS DEBUG and intra-program image
- Good formatting reduces cognitive load

© 2026, Robert Gezelter Software Consultant, All Rights Reserved

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

Session Notes & Materials:
http://www.rlgsc.com/openvms-bootcamp/2026/index.html

© 2026, Robert Gezelter Software Consultant, All Rights Reserved