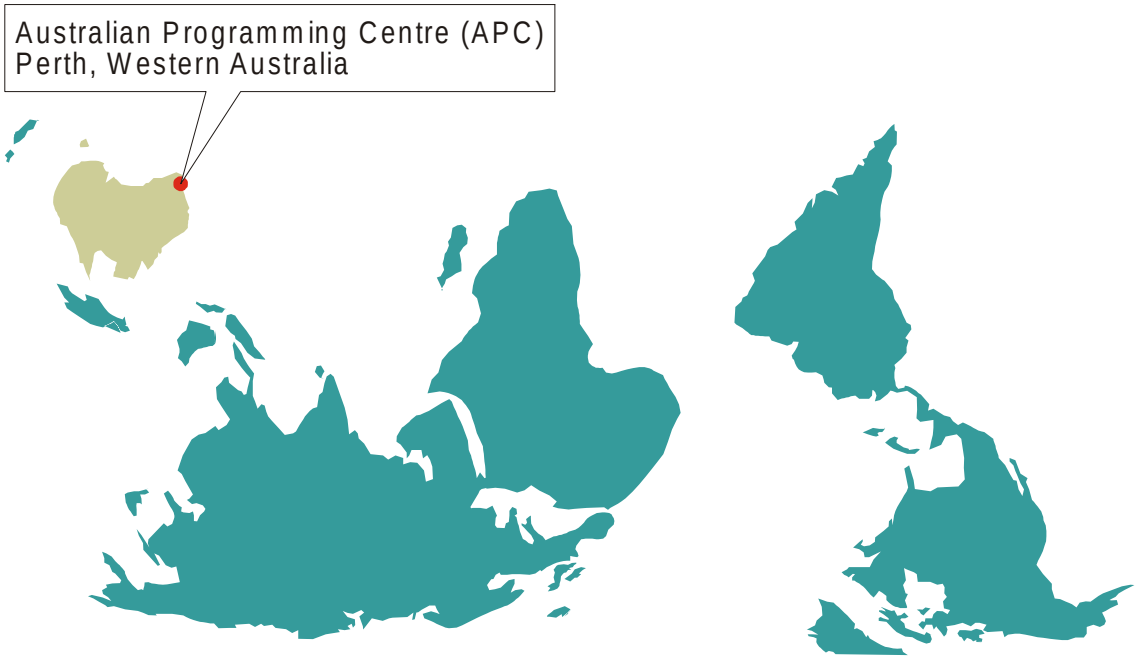
Developments Down Under



Presented for Jim Alexander, APC manager, by Alice Crema

2000 WAVV Conference
October 7-10, 2000
Colorado Springs

www.ibm.com/services/learning/conf/vmvse/

Contents

About the Australian Programming Centre (APC)

Completed APC development projects

- VS COBOL II 1.3.2 and 1.4.0 for MVS, VM and VSE/ESA
- C/370 run-time for VSE/ESA
- Language Environment for VSE/ESA (LE/VSE)
- PL/I for VSE/ESA (PL/I VSE)
- Debug Tool for VSE/ESA (DT/VSE)
- HLASM for MVS, VM, and VSE/ESA
- COBOL and CICS Command Level Conversion Aid (CCCA) for MVS & OS/390 & VM, and for VSE/ESA
- DITTO/ESA 1.3

Developments Down Under

Current APC service projects

- COBOL
- PL/I
- C/C++
- Assembler
- FORTRAN
- Pascal
- BASIC
- DPPX
- AOC
- SA OS/390
- DITTO
- SDF
- CCCA

- APC is Level 3 support for all products that it has developed
- Y2K Technical Support Centre, Asia Pacific (closed)

Developments Down Under

The Australian Programming Centre (APC) has been developing and servicing language products for the Santa Teresa Laboratory (STL) and other IBM software development labs since 1989. The APC team consists of IBM employees and experienced contractors drawn from the industry, combining intimate technical knowledge of internal IBM systems with first-hand customer experience. APC is based in Perth, Western Australia.

Jim Alexander, APC manager

VSE/ESA only

REXX for CICS on VSE/ESA

REXX for CICS on VSE/ESA

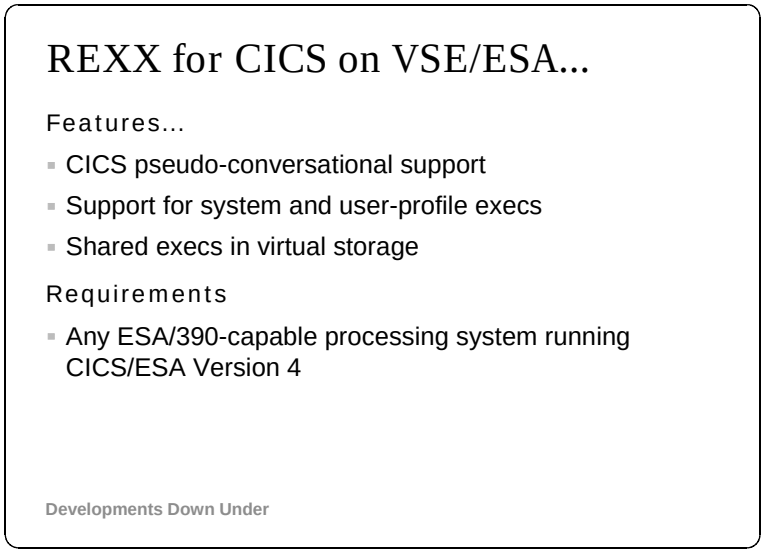
Interpreted language and application development environment

- Simple to configure and manage, simple to use editor (runs under CICS)
- No JCL, COBOL or CICS translator expertise required
- Ideal for:
  - Prototyping
  - Ad hoc systems programming (for example, to automate sequences of CICS activities)
  - Web-based programming

REXX for CICS on VSE/ESA...

Features

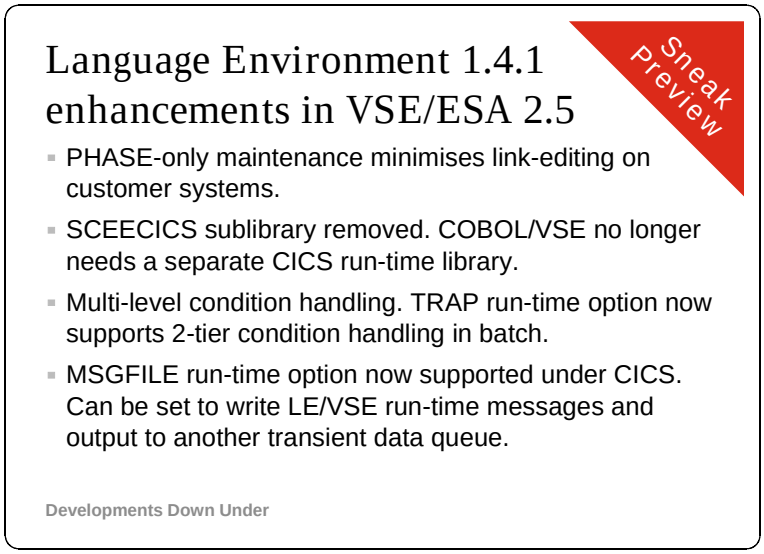
- SAA Level 2 REXX (non-stream I/O) language support
- SQL and EXEC CICS commands from REXX execs
- CICS native text editor (able to edit VSE Librarian members)
- High-level, VSAM-based file system
- High-level terminal I/O pane facility
- REXX interface to CEDA and CEMT transaction programs
- Support for subcommands written in REXX

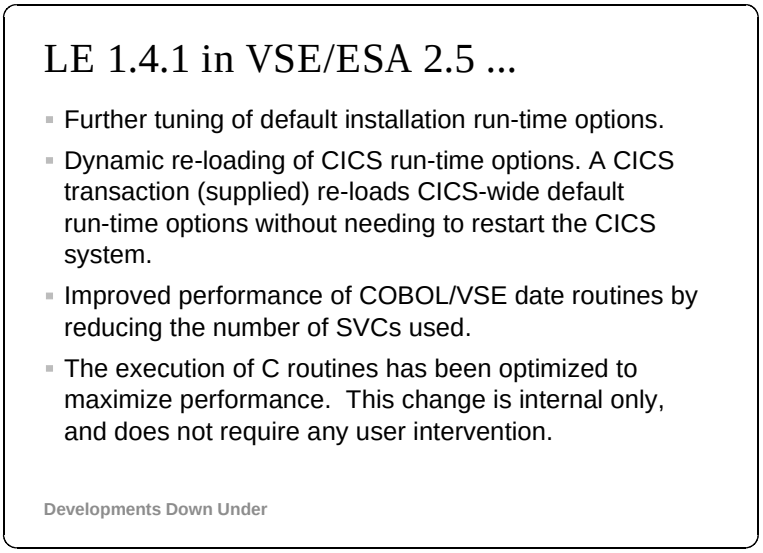


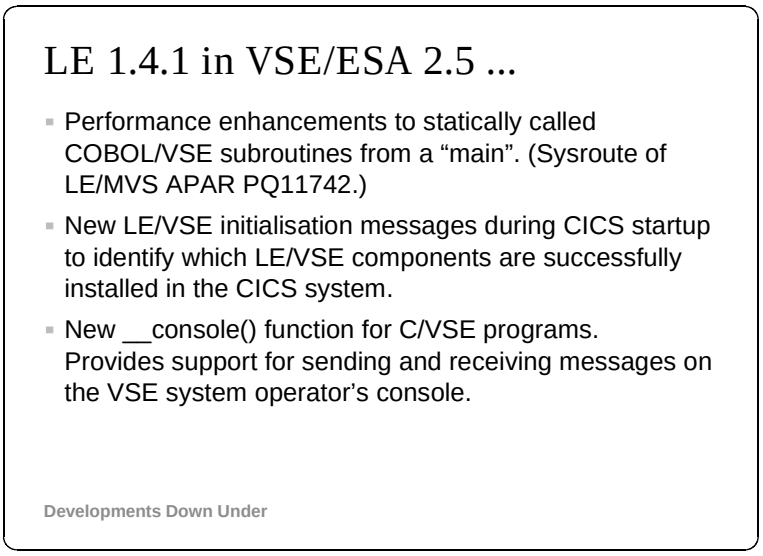
More information

- REXX for CICS on MVS/ESA announcement letter: www.ibmlink.ibm.com/usalets&parms=H_299-048
- REXX language home page: www2.hursley.ibm.com/rexx/

Language Environment enhancements in VSE/ESA 2.5







TRAP run-time option enhancements: MAX and MIN

TRAP specifies the level of condition handling that LE/VSE performs for user abends and program interrupts.

You must specify at least TRAP=((ON,MIN),..) in order for applications to run successfully.

TRAP=((ON,MAX),...) must be in effect for the ABTERMENC or ABPERC run-time options to have effect.

This option is similar to the STAE|NOSTAE run-time option currently offered by C/370.

The use of the CEESGL callable service is not affected by this option.

Syntax: TRAP=((ON|OFF,MAX|MIN),OVR|NONOVR)

IBM-supplied default: TRAP=((ON,MAX),OVR)

MAX (new for LE/VSE 1.4.1)

Instructs LE/VSE to activate full condition handling. This will involve the use of both STXIT AB and STXIT PC processing. If OFF is specified, then MAX has no effect.

MIN (new for LE/VSE 1.4.1)

Instructs LE/VSE not to use any STXIT AB processing for LE/VSE condition handling and to only use STXIT PC condition handling. This is required for internal LE/VSE failures that are part of application abend reporting and dump producing functions. If OFF is specified, then MIN has no effect.

Note: Each of the HLLs will still issue STXIT ABs as required regardless of the TRAP run-time option setting.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

Supplied NEWC CICS-wide run-time options facility

Supplied with LE/VSE 1.4.1 is the NEWC CICS transaction that allows activation of changed LE/VSE CICS-wide default run-time options, without the need to cycle the CICS system. This means that you can change your installation CICS-wide default run-time options by editing and running the supplied CEEWCOPT job, and then have these new run-time options activated, all while the CICS system continues processing online transactions.

This function is available for both CICS/VSE/TS 1.1 and CICS/VSE 2.3 systems.

You may change the transaction name used to perform this function to one of your choice. The program to execute is EDCCNEWC and the supplied NEWC transaction definition should be used as an example.

More information

- IBM VSE/ESA home page:
  www.s390.ibm.com/vse/
- Language Environment for VSE/ESA home page:
  www.s390.ibm.com/le_vse/

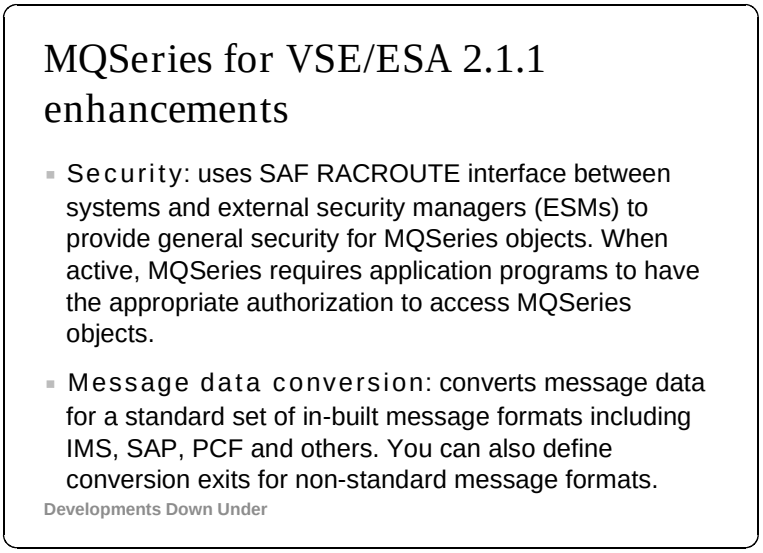
Related presentation

“Language Environment for VSE/ESA Hints & Tips”, Session E30
Presented by Wolfgang Bosch (Boeblingen Programming Laboratory, Germany)

This session updated for 2000 will contain more news and experiences with LE/VSE.

MQSeries for VSE/ESA 2.1.1 enhancements

IBM MQSeries for VSE/ESA provides a set of messaging and queueing services which support data transfer between distributed applications. These services allow applications to communicate without knowledge of the lower levels of the communications network and without specific knowledge of the location of other applications.



Security

VSE 2.4 introduced the SAF RACROUTE interface between systems and external security managers (ESMs). MQSeries/VSE 2.1.1 exploits this interface to provide general security for MQSeries objects including connections, queues and access authorities. MQSeries/VSE uses a standard subset of the RACROUTE interface which means it is not tied in to any particular ESM.

With 2.1.1, security becomes an installation option. When active, MQSeries requires application programs to have the appropriate authorization to access MQSeries objects. Secure application programs include CICS transaction programs, client connections and batch programs exploiting the MQ/VSE Batch Interface.

Message data conversion

Prior to release 2.1.1, MQSeries/VSE did not provide data conversion for message data. Data conversion was limited to internal MQSeries message headers. This was to allow MQSeries to communicate with other MQSeries systems running on different hardware, with different operating systems and potentially divergent codepages.

MQSeries/VSE 2.1.1 introduces codepage conversion for message data. It is now possible to configure MQSeries/VSE to convert message data for a standard set of in-built message formats including IMS, SAP, PCF and others.

It is also possible to define conversion exits for non-standard message formats. Codepage conversion for message data has also been extended to manage Unicode, DBCS, Euc and ISO.

MQSeries/VSE enhancements...

- Automatic VSAM space reuse: configurable, automatic reorganization of queues (VSAM KSDS) without affecting running applications. Minimizes downtime for applications requiring 24x7 availability.
- Java client applications (running on other platforms; not VSE) that use the Java Message Services (JMS) API can connect to and use MQSeries/VSE services.

Developments Down Under

Automatic VSAM space reuse

MQSeries/VSE uses VSAM for data storage. MQSeries/VSE queues are implemented as VSAM keyed sequential data sets (KSDS) which, in normal operation, require reorganization to maintain healthy performance. This can pose a problem for application systems that require 24x7 availability. MQSeries/VSE 2.1.1 introduces an automated reorganization mechanism for its VSAM data sets. This means MQSeries-dependant systems will not require “downtime” for scheduled VSAM reorganization. This mechanism is configurable, automatic and transparent to MQSeries/VSE applications.

Support for Java clients connecting to MQSeries/VSE

Java Message Service (JMS) offers developers an API for writing messaging applications in Java. JMS is available for MQSeries for AIX, HP-UX, Sun Solaris, and Windows NT.

Support for Java clients connecting to MQSeries/VSE will be introduced in 2.1.1. This means client programs written in Java running on supported platforms will be able to connect to and use MQSeries/VSE services.

MQSeries/VSE enhancements...

- TCP/IP: channel definitions can now use domain names in addition to IP addresses. Improved performance over TCP/IP connections.

Availability

- MQSeries/VSE 2.1.1 will be available in September 2000.

Developments Down Under

TCP/IP adaptions

Support for TCP/IP clients and connectability between MQSeries queue managers was introduced in MQSeries/VSE 2.1.0. The TCP/IP product for VSE has been improved and extended since its initial release. MQSeries/VSE 2.1.1 takes advantage of these improvements. In 2.1.1, channel definitions can use domain names in addition to IP addresses. Extensions to the Berkeley Standard Definition (BSD) socket interface have also allowed MQSeries/VSE to improve performance for TCP/IP client and queue manager connections.

Prerequisites

MQSeries/VSE 2.1 has the following prerequisites. All prerequisites should have the latest maintenance applied:

- VSE/ESA 2.3.1 (VSE 2.3.2 recommended)
- CICS 2.3
- LE 1.4
- TCP 1.3
- VTAM 4.2

More information

- Redbook: *MQSeries for VSE/ESA*, SG24–5647–00

  Available from:
  www.redbooks.ibm.com

  Acrobat PDF file:
  www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245647.pdf

- MQSeries Web site:
  www.ibm.com/software/ts/mqseries/

VSE/ESA and VM/ESA

HLASM Toolkit Feature: ASMPUT for Windows

The High Level Assembler Toolkit Feature Program Understanding Tool (ASMPUT) analyzes assembler language programs, and displays analyzed source code and the corresponding control flow graph.

You can use the control flow graph to trace complex control flows and inter-program linkages. The control flow graph is made up of nodes and arcs. A node corresponds to a group of lines of code, typically ending with a branch. An arc shows a connection between nodes; a jump, call or return from one line of code to another. (A node that is directly connected to another by an arc is a “linked node”.)

You can display different layers of the control flow graph. Higher layers display items in less detail, lower layers reveal items in greater detail. For example, when you expand a node by one layer, ASMPUT breaks the node holding many lines of code into a number of nodes holding fewer lines of code plus connecting arcs.

Apart from isolated nodes, you can trace a path from one node to another, moving along connecting arcs. The nodes immediately joined by arcs to a selected node are the nodes of most importance. Nodes further away are less important to the selected node. When you “remove the context” you remove the more distant nodes.

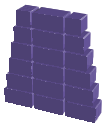
ASMPUT shows only the nodes directly related to the selected node. You can also add context, so that nodes related to those currently displayed will also be displayed. By adding or removing context, and by expanding or collapsing nodes, you can build a control flow graph that has the degree of simplicity and detail that you want.

Zooming in and out changes the size of the elements in the control flow graph, without making any difference to the structure of the graph. Clicking on a line of source code highlights the corresponding node in the graph. Likewise, clicking a node in the graph highlights the corresponding lines of source code. This means that you can trace the control flow either from the source code listing or from the control flow graph, using whichever you find the easiest. To prepare for using ASMPUT, you must create ADATA files for each program you want to analyze.

ASMPUT provides information from the High Level Assembler (HLASM) assembly of the programs, for example, listing all the assembly options. ASMPUT has three different windows. The Main window shows the ADATA files that are open, information about these files, and a source code listing. The Control Flow Graph window shows the control flow graph, and the options and icons you can use the change the structure of the graph. The Overview window shows a small copy of the control flow graph, and an area box for quick zoom and scroll control.

You can use ASMPUT on Windows or OS/2. The interface and processes of these two versions are essentially the same.

HLASM Toolkit Feature

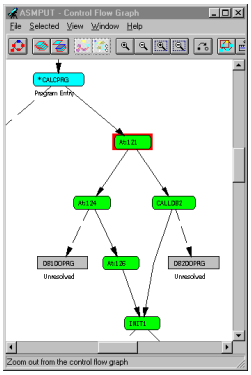


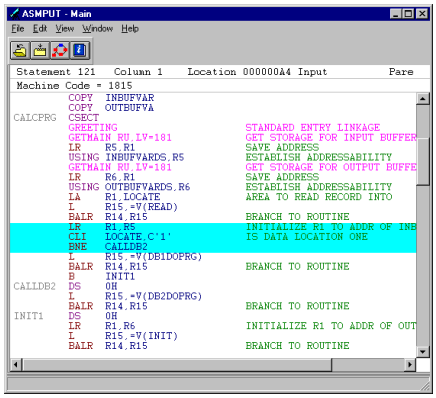
Program Understanding Tool (ASMPUT)
for Windows **(previous version was OS/2-only)**

- Analyzes assembler language programs and produces a control flow graph
- From this graph, you can trace complex control flows and program linkages
- 30-day evaluation version available on the Web
- Ask me for a diskette demo
- (Toolkit Feature also available under OS/390, MVS)

Developments Down Under

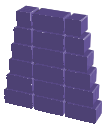
HLASM Toolkit Feature...





Developments Down Under

HLASM Toolkit Feature...



Other components of the
Toolkit Feature

- Interactive Debug Facility (IDF)
- Flexible disassembler
- Complete set of structured programming macros
- Cross-reference tool
- Comparison and search facility (Enhanced SuperC)

Developments Down Under

More information

HLASM Web site
(where you can download a 30-day trial version of ASMPUT for Windows):
www.ibm.com/software/ad/hlasm/

Related presentation

“IBM High Level Assembler Toolkit Feature: Program Understanding Tool (ASMPUT)”, Session E29
Presented by Clive Nealon (APC change team leader; for many products, including HLASM)

Come along and see how ASMPUT can help you analyze and debug assembler code.

VM/ESA only

DITTO/ESA 1.3 support for Exported Stacked Volumes (ESVs)

DITTO/ESA 1.3 support for Exported Stacked Volumes (ESVs)

- DITTO/ESA can now process ESVs created by the Export function of the Virtual Tape Server
- Two new DITTO/ESA functions for ESVs:
  - List (EVL)
  - Copy (EVC)

Developments Down Under

DITTO/ESA ESV support...

List (EVL) function

- Batch or online
- Short list
  - Lists all VOLSERs on the Exported Stacked Volume
- Long list
  - Lists all or only selected VOLSERs on the Exported Stacked Volume
  - Presents a volume layout for each VOLSER listed

Developments Down Under

Sample output from a short list:

```
DITTO/ESA for MVS
$$DITTO EVL INPUT=TAPE,TYPE=SHORT
Exported Stacked Volume Table of Contents for Volume VTS994
          VOLSER   SEQ NO
          EJ1015      1
          EJ1012      2
          EJ1019      3
          EJ1035      4
          EJ1021      5
          EJ1001      6
          EJ1029      7
          EJ1100      8
          EJ1099      9
          EJ0020     10
```

```
          EJ0007     11
EVL completed
$$DITTO EOJ
```

Output from a long list selecting only 4 VOLSERs:

```
$$DITTO EVL
INPUT=TAPE,TYPE=LONG,VOLSER=(EJ0020,EJ1019,EJ1029,EJ1035)
Exported Stacked Volume Table of Contents for Volume VTS994
          VOLSER   SEQ NO    Logical Volume Layout
          EJ1019      3      VOL1EJ1019
                             HDR1PE.MNT06.TAB351OO
                             ----- TAPE MARK -----
                                   Data File
                             ----- TAPE MARK -----
                             EOF1PE.MNT06.TAB351OO
                             ----- TAPE MARK -----
                             ----- TAPE MARK -----
                             ===== End of Volume =====
          EJ1035      4      VOL1EJ1035
                             HDR1PE.MNT12.TAB351OO
                             ----- TAPE MARK -----
                                   Data File
                             ----- TAPE MARK -----
                             EOF1PE.MNT12.TAB351OO
                             ----- TAPE MARK -----
                             ----- TAPE MARK -----
                             ===== End of Volume =====
          EJ1029      7      VOL1EJ1029
                             HDR1PE.MNT08.TAB351OO
                             ----- TAPE MARK -----
                                   Data File
                             ----- TAPE MARK -----
                             EOF1PE.MNT08.TAB351OO
                             ----- TAPE MARK -----
                             ----- TAPE MARK -----
                             ===== End of Volume =====
          EJ0020     10      VOL1EJ0020
                             HDR1.BACKTAPE.DATASET
                             ----- TAPE MARK -----
                                   Data File
                             ----- TAPE MARK -----
                             EOF1.BACKTAPE.DATASET
                             ----- TAPE MARK -----
                             ----- TAPE MARK -----
                             ===== End of Volume =====
EVL completed
$$DITTO EOJ
```

DITTO/ESA ESV support...

Copy (EVC)

- Copy up to 5 logical volumes in a single pass
- Each logical volume copies to a single physical volume
- Verifies requested volumes are on the ESV
- Copies performed in order requested
- Full logical volume copied (partial copy not supported)
- Copy to disk not supported
- New volume maintain VOLSER of output volume (original VOLSER maintained only if no VOLSER found on the output volume)

Developments Down Under

DITTO/ESA ESV support...

Availability

- DITTO/ESA for VM 1.3
  - PTF UQ40493
  - Available since February 2000
- DITTO/ESA for VSE 1.3
  - No support: VSE/ESA does not support the VTS hardware capabilities

Developments Down Under

More information

DITTO/ESA Web site:
www.ibm.com/software/ad/ditto/

VS FORTRAN 2.6 support for Additional Floating Point (AFP) registers

VS FORTRAN 2.6 Additional Floating Point (AFP) register support

- PTF introduces support for the 16 Floating Point registers in Generation 5 and Generation 6 S/390 Parallel Enterprise Servers
- Enables the compiler's Floating Point optimization (under control of a new compiler option) to use these additional registers where appropriate
- Can lead to faster execution of recompiled FORTRAN user applications

Developments Down Under

Ordering information

VS FORTRAN 2.6 program number: 5668-806

APAR number: PQ26305

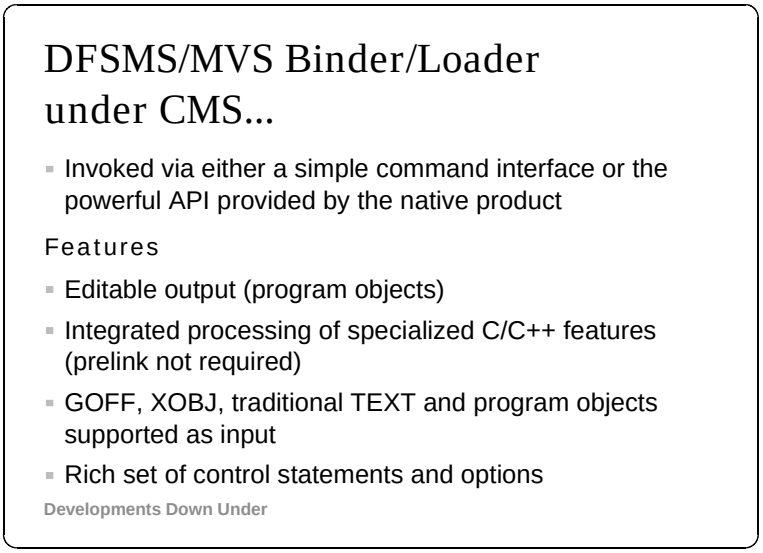
PTF number for VM/ESA: UQ33003

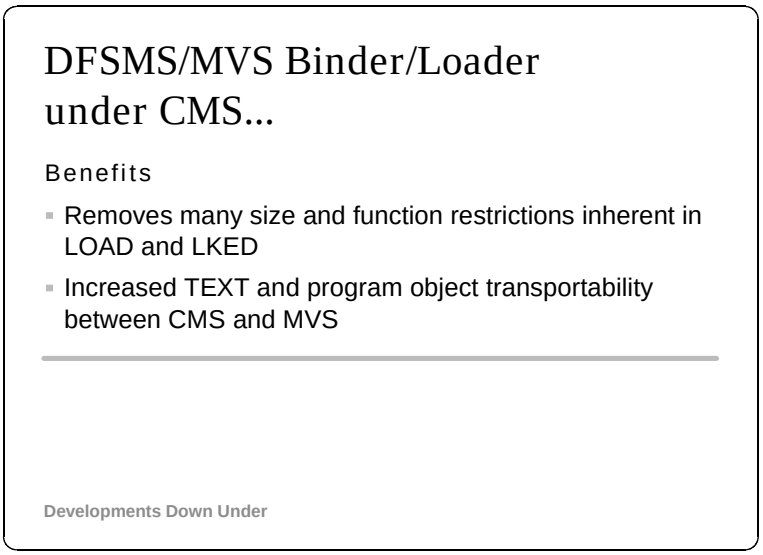
DFSMS/MVS Binder/Loader under CMS

DFSMS/MVS Binder/Loader under CMS

- Enables the DFSMS/MVS Program Management Binder and Loader in CMS
- Provides function similar to the LOAD / INCLUDE / GENMOD command sequence, and a superset of the function of the LE prelinker and the LKED command
- Produces an executable as a CMS MODULE file, a BFS file, or a LOADLIB member
- Not a port: the DFSMS/MVS binder is "cradled" to run unmodified in the CMS environment

Developments Down Under





Examples of using the BIND command from the CMS Ready prompt

This section presents a series of examples demonstrating the basic concepts and features of using the BIND command from the CMS ready prompt. In some cases the behaviour described may vary from what you would see on your own system. For example, if your installation defaults specify NO as the value for the CALL option, the external references examples would not operate as described.

Example 1: binding a single text deck with no external references

Assume you have a file called FILE1 TEXT on your A-disk containing object code produced by a language compiler. Entering:

```
bind file1
```

will bind the text file and produce a program object called FILE1 MODULE on your A-disk. A listing file called FILE1 SYSPRINT will also be produced.

Example 2: binding a single text deck with external references

Suppose FILE2 is another program with external references resolved by members of text libraries called MYLIB TXTLIB and SYSLIB TXTLIB. Entering:

```
bind file2
```

will result in error messages from the binder because no SYSLIB is available for autocall processing, and because of the unresolved external references. Entering:

```
filedef syslib disk global txtlib * (concat
global txtlib mylib syslib
bind file2 ( /nocmsauto
```

will bind the text file and resolve the external references to produce a program object called FILE2 MODULE on your A-disk. A listing file called FILE2 SYSPRINT will also be produced.

Example 3: specifying binder options

The CMS Binder command interface allows you to specify binder options either as “mapped” options, or as native binder options in an options string. The CMS system attribute is implemented using the CMS Binder API CMSSYSTEM option, which is mapped as the SYSTEM option, so if you also wanted to set addressing and residence mode attributes for FILE1, you could enter any of the following:

```
bind file1 (system amode 31 rmode any
```

```
bind file1 (/cmssytem=yes,amode=31,rmode=any
```

```
bind file1 (system/amode=31,rmode=any
```

Note: Options appearing before the slash (/) separator use a CMS-style syntax, while options appearing after the slash use binder JCL parameter string syntax described in *DFSMS/MVS Program Management*.

More information

DFSMS/MVS Program Management, SC26-4916-03
www.s390.ibm.com/bookmgr-cgi/bookmgr.cmd/BOOKS/DGT1M605/CCONTENTS?SHELF=EZ239116