IBM VSE/Enterprise Systems Architecture

Preparing a Product for VSE

Version 2 Release 4

SC33-6331-01

IBM VSE/Enterprise Systems Architecture

Preparing a Product for VSE

Version 2 Release 4

SC33-6331-01

Note !

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

Second Edition (June 1999)

This edition applies to Version 2 Release 4 of IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA), Program No. 5690–VSE, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Attn: Dept ECJ-BP/003D
6300 Diagonal Highway
Boulder, CO 80301
U.S.A.

or to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Deutschland Informationssysteme GmbH
Department 0215
Pascalstr. 100
70569 Stuttgart
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

Trademarks and Service Marks

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

ACF/VTAM
Advanced Function Printing
AFP
AS/400
AT
BookMaster
CICS
CICS/VSE
Common User Access
CUA
C/370
DB2
DisplayWrite
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ESA/390
ES/9000
IBM
IMS
Language Environment
MVS/ESA
MVS/SP
MVS/XA
OS/2
PR/SM
RETAIN
SQL/DS
System/370
VM/ESA
VSE/ESA
VTAM

About This Book

The purpose of this book is to further improve the satisfaction of VSE customers by facilitating a homogeneous, convenient and easy to install/easy-to-use program environment.

VSE is a package product with a particular structure and requirements customers are used to. Software vendors and IBM product developers who intend to provide programs to be installed and run under VSE should be familiar with the VSE environment. This will let their programs fit into the VSE environment and help meet customers' expectations.

Although almost all information provided herein is available from various other IBM sources (manuals and licensed materials), such sources are not always at hand and do not focus necessarily on those areas that may be of particular interest from a vendor's perspective. Therefore, this book shall serve as a selective, comprehensive and handy compilation of such relevant information. However, it is not intended to modify or replace the primary source referenced herein in any respect. In case of any doubt or ambiguity the primary source shall be used.

It remains solely vendors' business decision and responsibility if and to what extent their product implementation follows the structures described in this book. Of course, IBM must reserve the right to make changes according to its business judgement and needs.

IBM is interested in further improving this book. Readers who want to provide comments or suggestions are kindly asked to use the Reader's Comments Form enclosed.

VSE versions considered

The book takes into account all current versions of VSE.

How to Use This Book

The following list tells where you find information on the various topics in this book:

Part 1. IBM Contact Points

Chapter 1, IBM Communication Channels for Software Vendors
describes the various channels through which IBM offers support to vendors.

Part 2. Programming Interfaces

Chapter 2, Overview
tells which programming interfaces IBM recommends for use.

Chapter 3, Macros and Vendor Exits
describes the programming interfaces that IBM offers to vendors and how to use them.

Part 3. Documentation and National Language Support

Chapter 4, VSE Customer Documentation
describes how to design user-friendly software publications.

Chapter 5, Providing National Language Support (NLS)
describes what to consider when designing software for users speaking many different languages and conforming to different cultural conventions.

Part 4. Creating Installation Tapes and Servicing Your Product

Chapter 6, VSE Product Numbering Conventions
explains the IBM VSE product numbering conventions to avoid numbering conflicts at customer installations.

Chapter 7, Creating Installation Tapes
describes how to prepare a product distribution tape.

Chapter 8, Installing and Customizing Your Product
tells how to install a product from the distribution tape and customize it thereafter.

Chapter 9, Providing Service
describes how a product is serviced.

Part 5. Packaging and Service Samples

Chapter 10, Packaging of Products
presents sample job streams for packaging products.

Chapter 11, Library Member Types
lists the member types allocated for specific use.

Chapter 12, APAR Fix (ZAP)
gives an example of a ZAP.

Chapter 13, Programming Temporary Fix (PTF)
gives Program Temporary Fix (PTF) examples.

Chapter 14, Shipping PC Code with VSE
gives a sample of how to ship PC code with VSE.

Chapter 15, Job for Customizing
gives a sample of a customizing job.

Additional help is provided at the end of the book:

Appendix A, Bibliography
gives a detailed bibliography on VSE manuals.

The glossary
explains technical terms used in the book.

The index
helps you to locate information.

Summary of Changes

This manual has been updated for VSE/ESA 2.4 to reflect new and changed functions as follows:

- Changed vendor exit VSE/AF Console Support
- Changed vendor exit VSE/AF Supervisor (SUP and PSUP)
- Layouts of PRODEXIT DSECTs have been removed.

Part 1. IBM Contact Points

Chapter 1. IBM Communication Channels for Software Vendors

This chapter describes the channels through which IBM currently works with vendors:

- The toll free (U.S. only) information telephone lines established for vendors to get direct support on general communication.
- The IBM S/390 Developer's Association.
- The Early Test Program established especially for vendors to test and verify their applications at an early stage with the most current version of IBM software.
- The National Solution Center Database, which describes literally thousands of software programs from IBM and from vendors to IBM sales personnel.
- The VSE/ESA Software Newsletter.

VSE/ESA Home Page

VSE/ESA has a home page on the World Wide Web, which offers up-to-date information about VSE-related products and services, new VSE/ESA functions, and other items of interest to VSE users.

You can find the VSE/ESA home page at:

`http://www.ibm.com/s390/vse/`

Note: Because the communication channels outlined above are subject to review and change by IBM, please check with your local IBM representative for the actual status and applicable terms and conditions.

IBM Vendor Information Lines

IBM Software Vendor Information Line

The IBM Software Vendor Information Line is a toll-free 800 number that provides general information on IBM vendor activities and is meant for software vendors who do **not** have a specific IBM contact concerning their area of interest.

Prerecorded vendor message information: The 800-number prerecorded messages provide current information on IBM software vendor programs. Special focus is on those programs in which the content is subject to change. Please call this number when you have questions about the following:

- Submitting programs for IBM evaluation
- IBM hardware and software loaner programs
- IBM marketing selected non-IBM industry application programs
- Early access to selected IBM programs for use in the development process
- Technical support
- Seminars and technical education

Vendor questions: This toll-free number allows software vendors to ask specific questions on a variety of subjects. Questions cover **all** areas of interest to the vendor community, such as possible development relationships, IBM's marketing of the vendor's industry application program, SAA. All questions are reviewed daily by a subject expert. Most answers are given using phone within 24 hours.

Telephone numbers

Touch-tone phone **1-800-627-8363 (1-800-627-VEND)**

Rotary dial phone: **1-800-777-0979**

Dialing from outside the US may incur charges from the international portion of the call.

Note: This line is **not** intended to give you assistance for solving specific product problems.

VSE Vendor Information Line

The IBM Software Vendor Information Line and the S/390 Developers Association provide support for all IBM vendor activities. The S/390 Developers Association also provides VSE-related information covering the following:

- VSE vendor support offered at IBM's development Laboratories.
- Migration of vendor applications to VSE/ESA.
- Latest news on VSE/ESA.

Telephone and FAX numbers

To get more information, please use the following phone or FAX numbers, or send a note to the specified Internet IDs:

USA / Canada **1-800-627-8636 404-835-9900**

EMEA **country code for Germany+7031-16-2809**

INTERNET ID world-wide **S390DA@VNET.IBM.COM**

Vendors from Europe, Middle East, or Africa may also use these numbers:

Phone number **country code for Germany+7031-164082**

FAX number **country code for Germany+7031-163232**

Note: For assistance with VSE product problem resolution continue to contact your local IBM Support Center.

The S/390 Developers Association

Join the S/390 Developers Association!

It is an exciting program that offers many advantages for qualified developers who want to develop products either on VSE, VM, MVS or across the S/390 platforms. The program benefits its members by assisting them in obtaining the resources necessary to support diverse customer solutions across IBM's S/390 platforms. The offerings available through the S/390 Developers Association help you build and market solutions to meet your customers' needs.

Membership Overview

The S/390 Developers Association is a program that is available to companies that develop, or plan to develop commercially marketed software for the S/390 platforms (VSE, VM, and MVS).

Membership in the S/390 Developers Association is open to those companies that actively develop, or plan to develop commercially marketed software executing in an IBM S/390 environment under VSE, VM or MVS operating systems.

Electronic support for the S/390 Developers Association is provided through IBM's Conferencing Services.

For more information, see the IBM brochure *S/390 Developers Association*, G326-0545.

Early Test Program (ETP)

The Early Test Program provides software vendors with remote access to selected pre-General Availability (pre-GA) IBM products. These pre-GA products are made available, along with a host of supporting development and testing software, on IBM processors installed in Dallas, Texas, USA, and which are accessible throughout the world using the IBM Information Network. Software vendors use dial-up or leased line connections to access the IBM Information Network and the processors running the Early Test Program systems.

In addition, options are available for local tape distribution for installation on local vendor systems, and for access to publications and information only.

These Early Test Programs are open to all software vendors who wish to test their software for the purpose of porting, migrating, or regression testing their software in the new pre-GA environment. This provides software vendors with the opportunity to offer their products with a general availability that coincides with IBM products. Typical Early Test Program availability begins shortly after IBM announcement and concludes at product GA plus thirty days.

Support

Each participating software vendor is provided with sufficient resources to perform remote testing of their software in a pre-GA environment. Source code and macro libraries that can be made available with the product at GA, are also included on the Early Test Program host system. Typically each participant is given their own virtual machine (operating under IBM's Virtual Machine operating system), which provides the ultimate in flexibility for testing new code by allowing complete vendor control of their own virtual (simulated) processor, including console operations, unlimited accessibility to object code, and all required system authorizations (system key, supervisor state, etc.).

The Early Test Program has been certified as an Inter-Enterprise System Connection (IESC) that provides extensive security for isolation of software vendors and proprietary software products.

Documentation, education assistance (for selected programs), and technical support for pre-GA products under the Early Test Program is provided by the IBM Solution Developer Technical Support Center (SDTSC), with sponsorship by the product owner, IBM Boeblingen Programming Laboratory. Terms and conditions are defined in a supplement to the base IBM Information Network Agreement provided by the SVSC.

Charges

Based on the complexity of the VSE/ESA product introduction, and level of vendor participation in an Early Test Program, charges may be required to recover some the costs for running the program.

Charges for remote access programs are presently based on actual CPU usage (one hour minimum), with a one time charge, initiation fee and a low hourly rate, designed to encourage vendor participation. Vendors need only provide necessary hardware, software, and line charges to connect their equipment to the nearest IBM Information Network node. Charges for Local Tape Distribution are based on the expense of distributing the tapes and publications, and vary according to the program.

Further Information

Selected programs will be made available as new product releases are announced, along with the currently available programs. The Early Test Program is conducted by IBM Solution Developer Operations -- Solution Developer Technical Support Center located in Dallas, Texas. More information on current programs may be obtained from:

Address

IBM Corporation

Solution Developer Technical Support Center

Attn: ETP Program Advocate / 16-21-96

1507 LBJ Freeway

Dallas, TX 75234, USA

You can also call the program advocates at 1-800-553-1623 for more information regarding early access to selected IBM programs for use in the development process.

National Solution Center Database

IBM's National Solution Center Database is a constantly updated list of many applications, solutions, customer references, as well as IBM and non-IBM information representing over 15,000 solutions from IBM and vendors.

The database is available on IBM networks for IBM SEs and Marketing Representatives world-wide.

To get a listing of the software products and other information about any company that is currently listed on IBM's National Solution Center Database, or to get application forms to become part of the database, contact your local IBM Office or mail your request to:

Address

IBM National Solution Center

P.O.Box 2150

Atlanta, Georgia 30055, USA

In North America phone: **1-800-627-8363 (1-800-627-VEND)**.

Elsewhere contact your IBM Marketing Representative or dial IBM.

Example

Following is an example of an application description entered into the database and thus made available to IBM sales personnel:

DOCID X73321

TITLE Accounts Receivable

SYSTYPE
9370, 43XX, 30XX, ES/9000

SYSREQ DOS/VSE, VSE/SP, VSE/ESA, MVS/SP, MVS/XA or MVS/ESA with COBOL, CICS

VENDTYPE
IAS Authorized Industry Application Specialist

VENDOR

```
John Smith
100 E 100 St
New York, NY 100001
CONTACT   : Sue Smith
PHONE     : 212-555-5555
```

DESCRIPT

John Smith's Accounts Receivable System is an online, real-time, credit management system. The standard system provides the functions and features needed to support cash application for trade receivables and maintain real-time customer account information. A credit data base allows for extensive online inquiry of customer information, detail open item and mass open item payment selection. Credit management is optimized through online, real-time access to account data for maximum control of receivables. Combining John Smith's Accounts Receivable and Customer Order Processing Systems provides the greatest possible control of your company's receivables and credit exposure, while utilizing a common customer file and credit data.

Available options include automatic cash application, online help, extended credit checking, multi currency, automated credit agency interchange, capabilities for special order pricing, order consolidation, production and assembly orders, deals and promotional pricing, vendor-direct shipments, customer-entered orders, and Electronic Data Interchange (EDI) compatibility.

END USER DEFINITION:

Growth Accounts
Departmental Systems
Enterprise End User Systems
Large Data Center System

REVDATE
09/04/90

INSTCUST
10 - 24

REFER Contact vendor for references

IBMOFFIC
xxx xxxx xxxx

IBMPHONE
xxx xxxx xxxx

SUPPORT
At John Smith's, planning and installation of the applications system is a joint effort between John Smith's and the client. Implementation Assistance services provided include project planning and management, organization and detailing of project tasks, interface design and programming, data conversion assistance, specialized training programs, and onsite technical support. The Custom Package Modification services include design and specification preparation, programming, documentation preparation, and implementation. Once Implementation is complete, John Smith's Telephone Support Center provides 24-hour, year-round assistance. These services are provided at no additional charge as part of the standard six-month warranty; thereafter, with John Smith's Maintenance Agreement. Additionally, John Smith's provides separate training for user and system personnel. The instruction format is a combination of lecture and hands-on student interaction with the system(s). John Smith's consulting staff provides services beyond the data processing environment. Typical services might include assistance in defining a customer's needs, testing with customer data to provide insight into a system's potential, pre-installation planning and ongoing consulting to improve the utilization or effectiveness of the installed systems. Finally, John Smith's Services Division can help you extend the capabilities of our standard software to address your special requirements, whether they include extensive implementation support, project management, or customized enhancements.

```
COMMENTS
     Software available for cross-license              N
     Code available outside the United States          Y
     Marketing brochures available                     Y
     Demo available                                    Y
     Response line available                           Y
     Education available                               Y
     Defect correction available                       Y
     Modification/enhancement support                  Y
     Multi-licensing agreement                         Y
     Application copy protected                        N
```

CUSTRESP

```
**********************************************************************
* This description is based upon information obtained from the      *
* vendor, and is provided without independent evaluation or         *
* validation by IBM.  IBM, therefore, cannot ensure the accuracy    *
* of statements as to the functions, quality, or performance of the *
* vendor's offering and makes no warranties, expressed or implied,  *
* concerning the offering.                                          *
* You should contact the vendor for current information, including  *
* prices, terms, and conditions.                                    *
**********************************************************************
```

HARDWARE

John Smith application software products operate on all S/370 and S/390 processors (9370, 43XX, 30XX, ES/9000). The exact configuration of processor, DASD, and terminals depends on the volume of business transactions to be processed. Please contact John Smith for more information.

This offering is supported on the 9371 processor.

SOFTWARE

John Smith application software products are supported in the following system software environments:

Operating Systems DOS/VSE, VSE/SP, VSE/ESA, MVS/SP, MVS/XA, MVS/ESA

TP Monitor CICS

Data Storage VSAM, IMS/DB, DL/1, DB2, SQL/DS

Compiler COBOL

Utilities Sort/Merge

SAA

Uses SAA elements - YES
APPLICATION STRUCTURE - Stand-alone, Host with Nonprogrammable Workstation (NWS), Cooperative, Host with Programmable Workstation
COMMON USER ACCESS (CUA) - CUA 1989, Advanced Interface Design Guide, Graphical
ENVIRONMENT - OS/2 EE, MVS, CICS
COMMON PROGRAMMING INTERFACE (CPI) - Communications, Database, Presentation
COMMON COMMUNICATIONS SUPPORT (CCS) - 3270 DS, SNA Network Management, LU 6.2, Token Ring, X.25

Note: This product has been selected for inclusion in IBM's SAA catalog. For current information regarding SAA compliance, contact the vendor.

PRICE Contact vendor for prices.

TERMS A 99-year lease is standard for all John Smith products. Further, John Smith, Inc. grants a nontransferable and nonexclusive license to use the System to process the data of the division described in the Agreement. Upon the effective date of this Agreement, customers are invoiced 90% of the license fee, with payment due within 10 days. Included in the license fee is a six-month warranty that includes confirmation and correction of errors, and provisions for updates that are necessary for the System to continue to accomplish its principal functions. Additionally, John Soft, Inc. owns the rights to the software and guarantees that the software performs in accordance with the specification contained in the user and system documentation.

CURDESC

Document has been reviewed by the vendor within the last year.

The VSE/ESA Software Newsletter

The *VSE/ESA Software Newsletter* is published as a service to VSE technical people at customers, software suppliers, and consultants. It is distributed for free.

The *VSE/ESA Software Newsletter* includes information on product updates, describes major PTFs, and lists vendors and IBM products supporting the latest VSE/ESA version.

Address

Editorial office

IBM Deutschland Entwicklung GmbH
SW Market Planning 2
Department 3257
Schoenaicherstr. 220
D-71032 Boeblingen

Subscriptions

Please contact your local IBM and technical marketing representative to add, delete, or update subscriptions or give IBM a call.

Please contact the following phone or FAX numbers:

Phone number	**country code for Germany+7031-163099**
FAX number	**country code for Germany+7031-162575**

Your feedback, comments and suggestions to the newsletter will be appreciated.

Part 2. Programming Interfaces

Chapter 2. Overview

This chapter describes the different classes of interfaces to be considered when attaching a program to VSE and the problems arising from using attachments that are not IBM programming interfaces.

IBM Programming Interfaces

Only IBM Programming Interfaces are designed and officially released by IBM for the purpose of attachment. They are identified explicitly within the respective program manual belonging to the VSE System Reference Library.

In general, reasonable business considerations are taken to avoid changes that could affect user programs.

Other Attachments

Some vendors have attached their programs by using VSE source code or information from the IBM Diagnosis Reference manuals. These are n o t IBM programming interfaces designed and released for attachment. In addition, IBM advises against this attachment practice due to the following technical reasons:

- Source code is subject to constant change in the course of technical development. Thus, even a simple fix may render the attached program inoperative, not to mention the effect of numerous changes in new releases.
- In contrast to the programming interfaces described in the VSE Reference Library, the information given in the Diagnosis Reference library is intended for diagnosis purposes o n l y, that is to assist users in identifying and solving certain technical problems that may be encountered in the VSE environment; they are n o t intended for attachment.

  The macros documented in the Diagnosis Reference Manuals, for example macros developed for VSE/Advanced Functions, are:

  – Not documented as intended programming interfaces for customer use.
  – Not checked by VSE for interface use at the time of invocation. VSE assumes that the calling program comes in at the correct point in time providing the correct parameters.
  – Sometimes restricted in use in order to provide specific functions. If used otherwise, results are unpredictable.

Specifically, IBM discourages using a certain code sequence within the supervisor for attachment. Such use is even more critical when compared to the use of VSE/Advanced Functions' internal macros; those are intended to remain upward compatible whenever possible.

It must be clearly understood that problems resulting from such kinds of attachment are not within IBM's responsibility and are n o t solved by IBM.

Support of Multiple VSE Releases

Assume that you have an application that is running on different releases of VSE and it is accessing VSE control blocks directly. Assume also that a new release of VSE provides a programming interface that enables you to replace the old one. If this applies to your environment, then the application can make use of the new interface for the new VSE release and later ones, and still continue to run on the previous releases of VSE.

To allow for both kind of interfaces to coexist in your application, follow these steps:

1. Use the macro SUBSID INQUIRY to determine the level of VSE the program is running on. For a description of this macro, please see *System Macros Reference*
2. If the program runs on an older release of VSE that does not yet supply this interface, branch to the old code.
3. If the program is on the level of VSE providing this new interface, use the new interface at the highest possible level.

Vendor Support

IBM suggests that you make use of the following options for help regarding programming interfaces:

1. Use the normal channels to submit your requirement for a programming interface to IBM: the user groups at GUIDE, COMMON, or your local IBM representative. IBM will evaluate if those interfaces can be made available.
2. Make use of the Early Test Program to check out your program early. There you also have access to the supervisor lists and can acquaint yourself with changes. See “Early Test Program (ETP)” on page 5 for detailed information on the Early Test Program.

Chapter 3. Macros and Vendor Exits

Vendors can attach their applications to VSE using the officially supported interfaces, for example, the macros and the vendor exits. Using these interfaces, the vendor applications and VSE/ESA can communicate well, and this results in less work for testing vendor applications, and less effort for problem determination.

The official interfaces also increase the flexibility of IBM software development: IBM can improve things "below" these interfaces without having to be concerned about an impact on a vendor application.

Notice

This chapter contains Programming Interface Information and is an extract from the *Supervisor Diagnosis Reference Manual*, LY33-9164. We do not guarantee for the accuracy of this extract and will not accept any APARs concerning it. For details please refer to the named manual. If you have questions, please refer to the contacts in “IBM Vendor Information Lines” on page 3.

VSE/ESA 1.4 supports only the following exits: class=BAM, subclasses Pre-OPEN, Pre-CLOSE, Post-OPEN, and Post-CLOSE, and class=LNG. Starting with VSE/ESA 2.1 all exits are supported.

PRODID Macro - Accessing VSE Services

The PRODID macro allows software vendor programs to identify themselves to a VSE system and access authorized VSE services. Moreover, the macro allows to check which programs are already initialized in the system. Thus, one can check, for example, if an incompatible program is initialized.

An 8-byte token communicates the authorization. A job receives the token at DEFINE. The token becomes invalid at the deletion time, which is either through issuing PRODID DELETE or through the system shutdown.

The PRODID macro consists of the following five services:

- PRODID DEFINE to define your program to the system
- PRODID DSECT to get a description of the input and output of PRODID DEFINE and PRODID CHECK
- PRODID AUTH to obtain rights to use VSE services
- PRODID CHECK to search for products
- PRODID DELETE to cancel service access

For an example of how to use the PRODID macro in your program, see “Example of PRODID” on page 23.

PRODID DEFINE Service

The service is contained in the initialization routine of your program and notifies the system. PRODID DEFINE sends the program's “visiting card” to the system, which then honors the card with a token. The information on the visiting card, later on called notification entry, is kept by the system until it is removed by the program. The notification entries in the dump of the SVA help Service organizations to analyze a memory dump and identify those programs that might be involved in a certain problem.

The token given to the program functions as a key that enables the program to use a list of services. As a key to a room may be lent to other people, this token may be passed from a program to routines executing under a different program but supported by the token owner. For example, consider a program that supplies a particular access method to other programs. The access method is initialized in its partition, obtains the token, and deposits it in an area shared with the supported programs. Then the token is used whenever the routines of the access method (working for a supported program) need to be enabled for use of a service, or disabled again if no longer needed.

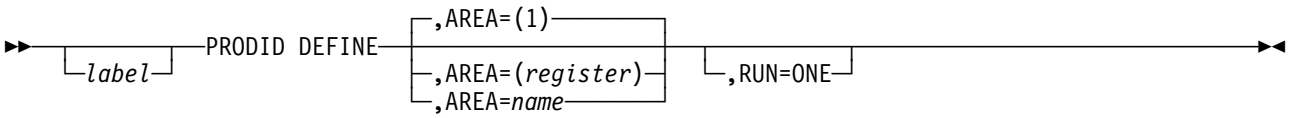
Prerequisites: In order to start a program that issues the DEFINE instruction, the following points should be considered:

- Users must have update right for the IJBVEND phase residing in IJSYSRS.SYSLIB. The phase IJBVEND is the part of the supervisor processing the PRODID request. Thus, the system administrator must have defined the user ID and the right in DTSECTAB. For information on how to update the DTSECTAB, see "Protecting Data" in the manual *VSE/ESA Guide to System Functions*
- The PRODID DEFINE service must be issued while the program is running in its own partition.

The token is deleted as the result of an explicit PRODID DELETE request or system shutdown, not with any end-of-task or end-of-job processing.

Requirements for the caller:

- Authorization: controlled by Access Control Facility
- AMODE: 24 or 31
- RMODE: 24 or ANY
- ASC-mode: Primary



Input Parameter Description:

AREA points to an area where the input information is available. The area can be specified in register notation (the default is 1) or as the name of the input area.

The area must be in an address range valid for this job.

The input for PRODID DEFINE, the text of the “visiting card” is a string described by the PRODID DSECT service. This input should be in character format so that no conversion are needed for printing the string. The input is accepted as passed along, except it must not be empty; that is, none of the three fields IJBCOMPN, IJBPRODN, IJBVRM can be binary zero. In such a case the request would be rejected and a return code issued.

The fields for identifying the company, the product, and the release level should contain the following information:

IJBCOMPN The name of the company owning the product and issuing the DEFINE request. Only *one* meaningful abbreviation should be used for all products of a company.

IJBPRODN The name of the product. Only *one* meaningful abbreviation should be used for all different releases of a product.

IJBVRM The version, release, and modification level of the program.

The information in these fields may have a different length than the one specified by the DSECT. It may, however, not use more than the combined length of the fields IJBCOMPN to IJBVRM. For example: say you would want to indicate a release level that requires more than the 6 bytes allotted to IJBVRM. In this case, the release level must start somewhere in IJBPRODN so its end coincides with the end of IJBVRM.

Note: The CHECK function does not support such overflowing information, but treats, for example, the field IJBCOMPN as containing the company name and nothing else.

RUN=ONE The program runs only once per system as, for example, VSE/POWER.

Output: is an 8 byte token returned in the input area at label IJBTOKEN. Note that the token is associated with the input string, not with a task or partition.

On return, the contents of register 0 is destroyed.

Return Codes: returned in register 15

0 The system accepted the input and returned a token.

4 The same string is already defined for the system or partition, the same token was returned as previously.

8 Maximum of 256 products already defined.

12 Control block format error. At least one of the fields identifying company, product, or release level is binary zero, or IJBVIDL is incorrect.

16 IJBVEND not loaded.

20 GETVIS error, return code from GETVIS in register 0.

Cancel Conditions:

X'0B' Security check failed

X'21' An unknown function code is passed to PRODID (corrupted macro expansion).

X'25' Invalid address of parameter list or parameter

PRODID DSECT Service

This service generates a DSECT describing:

- Input expected by PRODID DEFINE
- Output returned by PRODID DEFINE
- Input expected by PRODID CHECK.
- Output returned by PRODID CHECK.

For information of the value of these fields, see page 16 and “PRODID CHECK Service” on page 20.

The layout is:

Figure 1. PRODID Parameter List

Field Name	Description	Field Type & Length
IJBVIDL	LENGTH OF INPUT AREA	AL2(IJBFINST)
IJBVIFL1	FLAGBYTE, RESERVED	CL1
IJBVIFL2	FLAGBYTE, RESERVED	CL1
IJBCOMPN	NAME OF COMPANY, PROGRAM OWNER	CL14
IJBPRODN	NAME OF PROGRAM	CL16
IJBVRM	VERSION AND RELEASE OF PROGRAM	CL6
IJBTOKEN	TOKEN RETURNED BY DEFINE	CL8
IJBCKLEN	LENGTH OF OUTPUT AREA FOR CHECK	CL2
IJBFINST	IJBVIDL	*
IJBCKARE	BEGIN OF OUTPUT AREA FOR CHECK	*
IJBCOMPO	NAME OF COMPANY, PROGRAM OWNER	CL14
IJBPRODO	NAME OF PROGRAM	CL16
IJBVRMO	VERSION AND RELEASE OF PROGRAM	CL6
IJBTIDO	TASK ID RETURNED BY CHECK	CL2
IJBPIKO	PARTITION ID RETURNED BY CHECK	CL2

Input Parameter Description: None.

Output: None.

Return Codes: None

PRODID AUTH Service

The PRODID AUTH service enables a program with a valid token to obtain the right to use the services or to give up this right. The following services are available through PRODID AUTH:

- EXTRACT ID=PART
- EXTRACT ID=SVA
- GETFLD FIELD=ALET (add to DUAL of current task)
- MODFLD FIELD=PASCOPE1
- MODESET (since VSE/ESA 2.1.0)
- PRODEXIT (since VSE/ESA 2.1.0, 1.4 retrofit for BAM and LNG)
- TREADY COND=ICCF
- TREADY COND=NO
- SYSDEF
- XMOVE (since VSE/ESA 1.3.1)
- MODESET

The system uses a fast path to process this request. The right is granted to the task issuing the request, and is removed at end-of task if not already revoked. This need not be the same task that notified the system with PRODID DEFINE and obtained the token.

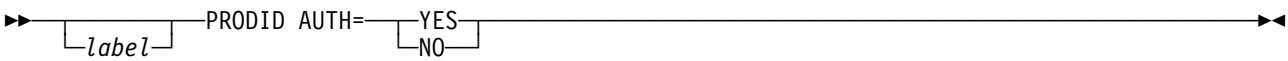
PRODID AUTH gives non-IBM software - and even an IBM program not included in SUBSID support - a status comparable to an IBM subsystem as, for example, CICS. This also means the programs must use utmost care when using the services listed below. The checking in these services is not as extensive as required for a user interface.

WARNING: ANY ERROR IN CALLING THE SERVICES, FOR INSTANCE WITH INCORRECT PARAMETERS OR AT ANY WRONG POINT IN TIME, MAY DAMAGE THE SYSTEM.

Note: Each AUTH=YES request increases a one byte counter attached to the issuing task, each AUTH=NO request decreases it by one.

Requirements For The Caller:

- Authorization: valid token
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary or AR



Input Parameter Description:

Registers 1 and 2 must contain the token previously obtained by PRODID DEFINE

AUTH YES requests the right to use special services. If the token passed in registers 1 and 2 is valid, this right is granted.

NO gives up the right.

Output: None.

On return, register 0 is destroyed.

Return Codes:

passed back in register 15

0 Right granted for AUTH=YES. Right set off for AUTH=NO.

4 Right revoked for AUTH=NO, but was already revoked or never requested.

8 Right not granted, more than 255 consecutive AUTH=YES requests for this product within this task.

16 IJBVEND not loaded.

Cancel Conditions:

X'21' Invalid token or an unknown function code is passed to PRODID (corrupted macro expansion).

PRODID CHECK Service

PRODID CHECK searches the accumulated notifications entries for either a certain known product, or all releases of such a product, or all products of a certain company, and returns any notification entries found. Also, notification entries of all programs may be retrieved; this is useful when determining the cause of a problem.

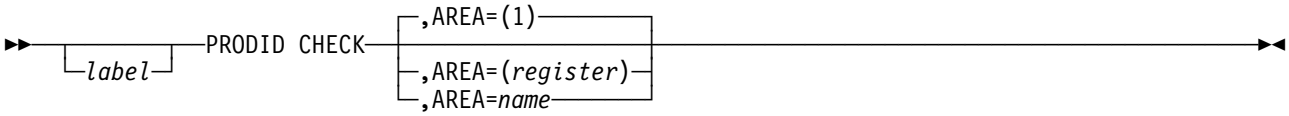
With this service the program checks for a known string.

All occurrences found are moved to the user's area starting at IJBCKARE if the length of this area is sufficient. If the length is not sufficient, a return code indicates this fact and the total number of matching entries is returned in register 0. Then the request may be repeated with a larger area.

Requirements for the caller:

- Authorization: none
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary

Invocation:



Input Parameter Description:

AREA points to the area that holds the search information. The area can be specified in register notation (the default is 1) or as the name of the input area.

The area must be in an address range valid for this job.

The layout of this string is described by PRODID DSECT. The following values may be passed as input:

IJBCOMPN contains the name of the company owning the product that is looked for.

If this field contains binary zeros, all notification entries are returned.

IJBPRODN contains the name of the product looked for. If this field is set to binary zeros, all notification entries with the same company name are returned.

IJBVRMN contains the version, release, modification level of the program checked for. If this field is set to binary zeros, all notification entries with the same company name and product name are returned.

IJBCKARE is the begin of the output area, it is supplied in the input pointed to by AREA. In this area the matching notification entries are moved provided they fit.

IJBCKLEN specifies the length of the output area. The output area must be in an address range valid for this job.

Output: The part from IJBCOMPN to IJBVRM of the found notification entry or entries - including the TIK and PIK of the partition where PRODID DEFINE was issued first - is moved to the area starting at field IJBCKARE in the length specified in IJBCKLEN. If the defining task terminated in the meantime, the fields for TIK/PIK are set to binary zeros. See also “PRODID DSECT Service” on page 17 for the layout of one returned notification entry.

On return, register 0 contains the number of entries found.

Return Codes:

codes passed back in register 15

0 Entries matching the search criteria were moved to user area.

4 Entries matching the search criteria were moved to user area. But there are more matching entries that could not be moved because the output area was full. Register 0 contains the number of all matching entries, so repeat request with larger area.

8 No match found or no entry existing.

12 Control block format error (for example, incorrect IJBVIDL).

16 IJBVEND not loaded.

Cancel Conditions:

X'21' An unknown function code is passed to PRODID (corrupted macro expansion).

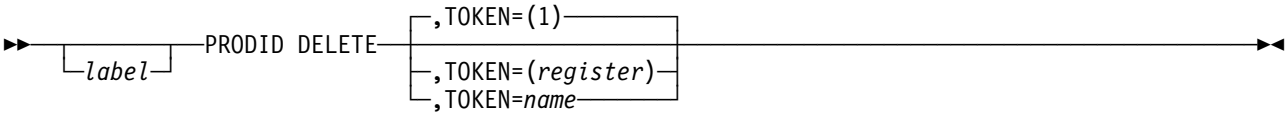
X'25' Invalid address of parameter list or parameter.

PRODID DELETE Service

With this service the program requests to delete its notification entry and invalidate its associated token. This should be used in the program's termination routine.

Requirements for the caller:

- Authorization: valid token
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary



Input Parameter Description:

TOKEN points to the area that holds the token that was previously obtained by PRODID DEFINE. The area can be specified in register notation (the default is 1) or as the name of the input area.

The area must be in an address range valid for this job.

Output: None

On return, register 0 is destroyed.

Return Codes:

passed back in register 15

0 Entry deleted. Authority was already reset before DELETE request was issued.

4 Entry deleted, but authority was still granted. Authority is removed.

8 Entry deleted. Authority was already reset before DELETE request was issued. At least one PROEDEXIT DEFINE was removed.

12 Entry deleted, but authority was still granted. Authority is removed. At least one PRODEXIT DEFINE was removed.

16 IJBVEND not loaded.

Cancel Conditions:

X'21' Passed token was invalid. This could be caused by a second DELETE request. Or an unknown function code is passed to PRODID (corrupted macro expansion).

X'25' Invalid address of parameter list or parameter.

Example of PRODID

Assume an application that needs to use one of the VSE services. The following example illustrates how you might use PRODID in your program.

Defining your Program to VSE and Requesting a Token: In the initialization routine define your application to the system with the PRODID control block MYPROGID and the following code sequence:

```
         ...
* define your program to VSE and ask for a token
         LA    R1,MYPROGID        point to your program's PRODID
*                                 control block
         PRODID DEFINE,(1)        pass its address to VSE with your
*                                 DEFINE request
         LTR   RF,RF              check for return code
         BNZ   CHECKRC1           go analyze return code
* VSE put a token into your PRODID control block
         ...
```

Figure 2. Requesting a PRODID Token

```
MYPROGID DC    AL2(IJBFINST)       length of control block defined
*                                  by VSE
         DC    X'0000'             flag bytes
         DC    C'ANY S/W CORP.'    Company Name used with all products
         DC    C'ANY PROGRAM NAME' Name of this program
         DC    C'020100'           version, rel., mod.level of above
*                                  program
         DC    CL8' '              token passed back by VSE for my
*                                  program
         DC    X'0000'             no area for check needed
MYPROG   PRODID DSECT
```

Figure 3. Coding a PRODID Control Block

Using a VSE Service: Using a VSE service includes the following three steps:

1. Obtain the authority to use the service
2. Use the service
3. Return authority no longer needed.

   Note: It is safer to run with the lowest possible level of authority.

```
         ...
* obtain right to use a service restricted to special users
         LA    R1,MYPROGID        point to your program's PRODID
*                                 control block
         USING MYPROG,R1
         LM    R1,R2,IJBTOKEN     and load your token into r1/r2
         DROP  R1
         PRODID AUTH=YES          pass it to VSE asking for authority
         LTR   RF,RF              check for return code
         BNZ   CHECKRC2           go analyze return code
         ...
* execute the service needed
         ...
* return authority
         LA    R1,MYPROGID        point to your program's PRODID
*                                 control block
         USING MYPROG,R1
         LM    R1,R2,IJBTOKEN     and load your token into r1/r2
         DROP  R1
         PRODID AUTH=NO           return authority
         LTR   RF,RF              check for return code
         BNZ   CHECKRC3           analyze return code
         ...
```

Figure 4. Using PRODID AUTH

In your termination routine delete your program's entry to leave a clean system.

```
         ...
* remove your program's PRODID information from VSE
         LA    R1,MYPROGID        point to your program's PRODID
*                                 control block
         USING MYPROG,R1
         LM    R1,R2,IJBTOKEN     and load your token into r1/r2
         DROP  R1
         PRODID DELETE            ask VSE to invalidate the token and
*                                 to remove your PRODID information
         LTR   RF,RF              check for return code
         BNZ   CHECKRC4           analyze return code
         ...                      continue with your clean up
         ...
```

Figure 5. Deleting the PRODID Token

PRODEXIT Macro - Handling Vendor Exits

This section describes the handling of the PRODEXIT services and the vendor exits. The services are described in detail starting with “PRODEXIT DEFINE Service” on page 29, the exits in “Vendor Exits” on page 37.

Exit Specification

A set of vendor interfaces are provided via exit points that must adhere to the following specifications:

1. To use the vendor exits, the application must be authorized. The valid token is obtained from the PRODID macro, the user must be authorized (DTSECTAB) to access IJBVEND and the products exit routine.
2. Exits can be specified for update (for example, DTFs, or JCL statements) or monitoring (no update). Monitoring exits receive control after exits defined for update. Several exits can be specified for a class or subclass, however only **one** exit per valid PRODID token. Exception: Monitoring and update exits for the same class or subclasses can coexist for the same TOKEN. The exits are activated in FIFO or priority order (depending on PRODEXIT DEFINE specification).
3. Addressability to the work area has to exist for every possible task under which an exit can be invoked. The exit routine itself has to be within the SVA (high or low). Starting with VSE/ESA 2.1, AMODE must be 31. Reentrancy has to be considered.
4. An exit can be enabled/disabled without affecting other exits for the same event type.
5. The different exit specifications are stacked per event and invoked as needed for the specific exit.
6. Vendor exits that are elegible for the dynamic ENABLE/DISABLE service can temporarily be enabled/disabled from the vendor in his vendor dispatcher exit routine.
7. Vendor exits have to consider that SVCs in ICCF pseudo partitions are intercepted by ICCF, which may change the input to a specific device.

Classes and Subclasses

The exits can be defined for given exit points in VSE/ESA components. A VSE/ESA component is called a **class**, the exit point **subclass**.

Classes are SUP, PSUP, SVC, FSVC, BAM, AIT, DOC, LNG, DOCP.

Subclasses can be defined for every class, for example, a subclass for class SUP is the POSTFCH exit, for class BAM the PRE-OPEN exit. If no subclass is specified, all existing subclasses of the specified class are defined automatically.

Note: PRODEXIT services for some subclasses of the classes BAM and LNG are also available in VSE/ESA 1.4 or in VSE/ESA 1.3 including the PTF for APAR DY43436.

Exit Process

The activation service PRODEXIT ACTIVATE can be used by VSE/ESA components at given points and transfers control to defined vendor exits. Depending on the environment, PRODEXIT ACTIVATE expands:

- In supervisor state to a BASSM call to the ACTIVATE processing (if no exit is enabled, control is returned immediately).
- In problem program state to a normal SVC.

Vendor Exit PSW Key: The vendor exit will be activated with the same PSW key as the issuing subclass of the exit PRODEXIT ACTIVATE.

PRODEXIT Communication Area Location At Exit Entry: The location and storage key of the PRODEXIT communication area, which is input for the exit entry, is described in the corresponding class/subclass paragraph.

An eight byte user area (IJBVUSW) may be given as input to the PRODEXIT DEFINE service for each exit. This field can hold, for example, an anchor for a work area and is given to the exit during activation.

- The PRODEXIT ACTIVATE service returns after **all** vendor exits of same class/subclass are processed.

  Note: Exceptions are described in the corresponding chapter of the vendor exit definition.

If more than one vendor exit is established for a subclass, the priority defined (during PRODEXIT DEFINE in the PRODEXIT area) is used to determine the activation order. Exits defined with the same priority will be activated in the sequence the PRODEXIT DEFINE was issued (FIFO).

Vendor exits defined for **update** (via PRODEXIT DEFINE) are activated prior to the **monitoring** exits. Within the 'update' and 'monitoring' exits the priority is used to determine the activation sequence. 'Update' exits are processed before 'monitoring' exits. 'Monitoring' exits are, per convention, **not** allowed to change any system information, a check however is not possible.

If more than one exit is to be activated, the next exit will get the output (if any) of the former exits within the same type. Exceptions are described in the class/subclass.

Note: If there are more exits defined for **update**, the one with the highest priority will be invoked first. The one with the lowest priority will be invoked last and therefore this one has the final chance to change the output before the control will be given back to the 'ACTIVATE' component.

The DEFINE, DELETE, and CHECK services expand to normal SVCs, the ENABLE and DISABLE services to fast path SVCs. The ACTIVATE and RETURN for PSTATE services also expand to normal SVCs, but the ACTIVATE and RETURN for SSTATE services result in BSM or BASSM. The VSE/ESA 2.2 dynamic ENABLE/DISABLE services will be invoked via BASSM.

Exit Scopes

Every exit has a **scope**. When the class or subclass is enabled for:

system scope	the exit is activated for all tasks.
partition scope	the exit is activated for all tasks within the partition, including system tasks that execute services for this partition.
task scope	the exit is activated for the current task only.

Note: Whether an exit is activated or not can be overruled dynamically by the new ENABLE/DISABLE services.

Whenever a new task is attached, it is automatically ENABLEd for:

- An exit of **system scope** if one was enabled.
- An exit of **partition scope** if the new task belongs to the partition for which an exit was enabled.

Note: An attach will fail if the necessary control blocks for vendor exit activation cannot be GETVISed. This can happen if there are vendor exits defined in the system that are 'flagged' to be critical ones

(see PRODEXIT DEFINE). In the case of dynamic partitions, the allocation is done anyway, but a message is issued to the console.

Exit Types

There are two types of exits, SSTATE and PSTATE.

Supervisor State (SSTATE) Exits: The exits receive control with the RID (the RID identifies the environment in which the task is running, for example, the usage of save areas) as described in the corresponding class/subclass paragraph (RID ¬= USERTID) and are disabled for interrupts. Register 1 points to a parameter list holding exit-related information (described by “PRODEXIT DSECT Service” on page 36). Register 15 holds the exit routine address at exit entry and may be used by the exit routine. The exit has to return with PRODEXIT RETURN. The exit will receive control in primary mode and AMODE 31. The high order bit of the exit address (defined by PRODEXIT DEFINE) has to be set. General purpose and access registers are saved before activation of the exit and restored after PRODEXIT RETURN. All areas as well as the exit routine itself have to be fixed for RID = SYSTEMID and may be pageable for environments, that allow page faults (RID=REENTRID).

Note: With VSE/ESA 2.4 the space-switching program call (PC-ss) is supported. Therefore, when a program gets control after an interupt, the primary space is not necessarily the allocating space.

Problem Program State (PSTATE) Exits (RID = USERTID): The PRODEXIT ACTIVATE service provides a save area and saves the status of the service issuer (PSW, general purpose register and access register). The vendor exit receives control in primary mode, problem program state (RID = USERTID) and AMODE 31. High order bit of the exit address (DEFINE service) has to be set to 1 if running in ESA environment. Register 1 points to a parameter list holding exit-related information (described by PRODEXIT DSECT). Register 15 holds the entry point address. The exit has to return via the PRODEXIT RETURN service, which may schedule another exit or restore the saved registers and return to the calling component.

Notes:

1. PSTATE exits are interruptible, however, external interrupts that drive IT, TT, and OC exits are delayed.
2. Per task one PSTATE exit, one SSTATE exit with RID=REENTRID, one SSTATE exit with RID=SYSTEMID (excluding FSVC CLASS), and one SSTATE exit with RID=SYSTEMID CLASS = FSVC can be active the same time.
3. A program check in the vendor exit or an abnormal termination causes skipping of normal program check exits or abend exit routines.
4. Any STXIT macro invocation of a vendor exit leads to ERR21 ==> 'illegal SVC'.

Register Conventions

On Exit Entry:

register 1 address of the area as defined by PRODEXIT DSECT
register 14 exit return address
register 15 exit entry point address

There is no need to save and restore registers for the user of the PRODEXIT ACTIVATE service. However, in case of SSTATE ACTIVATE out of the SVA, the ACTIVATE issuer has to provide a save area via R13.

On PRODEXIT Service Return: On return from any PRODEXIT service, the standard register convention applies, that is, the original contents of R0, R1, and R15 might be destroyed, R14 must have the same value as on entry to the exit.

Deleting an Exit

It is recommended to delete an exit (PRODEXIT DELETE), when the exit is no longer needed. If no deletion occurs, the enable time depends on the scope of an enabled exit:

system scope	enabled until next IPL.
partition scope	enabled until de-allocation of partition. (1.)
task scope	enabled until completion of end-of-task process. (1.)

Notes:

1. If several partitions/tasks are enabled, only the ending partition/task is DISABLEd; any exit remains being DEFINEd until the next IPL if not explicitly deleted via the PRODEXIT DELETE service.
2. An exit eligible for dynamic services remains enabled for dynamic services until IPL or until it is explicitely deleted.

Recovering from Errors

In case of abnormal termination of an exit, the dump routine might issue a cancel message and the corresponding failing address indicates that a vendor exit caused the problem. Other exits of the same subclass are skipped.

An abnormal termination of vendor exits running with RID=SYSTEMID leads to a hardwait.

PRODEXIT Services

The PRODEXIT macro provides the following services:

- “PRODEXIT DEFINE Service” on page 29.
- “PRODEXIT ENABLE Service” on page 32.
- “Dynamic PRODEXIT ENABLE” on page 33.
- PRODEXIT ACTIVATE activates an exit in the corresponding components (vendor exit hooks). This service runs automatically for enabled exits, and is described in PLM documentation (SY33-9164).
- “PRODEXIT RETURN Service” on page 33.
- “PRODEXIT DISABLE Service” on page 34.
- “Dynamic PRODEXIT DISABLE” on page 35.
- “PRODEXIT DELETE Service” on page 36.
- “PRODEXIT DSECT Service” on page 36 shows the contents of the input/output areas of the PRODEXIT services. These areas are called **PRODEXIT area** and **PRODEXIT Extension area**.

The New Dynamic Enable/Disable Services: The VSE/ESA 2.2 dynamic PRODEXIT ENABLE/DISABLE services allow vendors to improve the performance of vendor exits especially in a VSE/ESA multiprocessor environment. This can be achieved by exploiting the VSE/ESA 2.2 Vendor Dispatcher Exits. These exits are running as dispatcher extensions in supervisor mode and are *parallel code* in terms of the turbo dispatcher terminology. Within the Vendor Dispatcher Exit, the vendor can decide whether he wants to have the normal vendor exit, as defined via PRODEXIT DEFINE, being invoked for that instance or not. This adds some extra code, but it is parallel code and can be executed on multiple processors at the same time. The normal vendor exits are non-parallel code, and by avoiding unnecessary invocations of them, the ratio of parallel code to non-parallel code improves, and so does the overall performance in a multiprocessor environment.

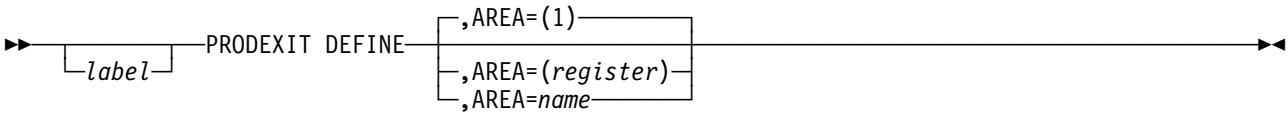
PRODEXIT DEFINE Service

PRODEXIT DEFINE defines a vendor exit of a given class and subclass. If the subclass is omitted, the exit is defined for all subclasses. The DEFINE service returns a PRODEXIT token that allows to use PRODEXIT ENABLE, DISABLE and DELETE services. Before an exit can be activated, it has to be enabled using PRODEXIT ENABLE; initially, the exit is disabled if it is not defined as being eligible for dynamic ENABLE/DISABLE services.

Vendor exits **must not** issue PRODEXIT DEFINE.

Requirements For The Caller:

- Authorization: valid PRODID token (as returned by PRODID DEFINE)
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: SVC



Input Parameter Description:

AREA= points to the PRODEXIT area where the input information is available. It may be specified in register notation or as the name of the input PRODEXIT area. The PRODEXIT area must be in an address range valid for this partition. The following fields of the PRODEXIT DSECT are used as input parameters:

IJBVCLS class of exit. The content is described by PRODEXIT DSECT. Only one exit class can be specified. Class is required.

IJBVSCL subclass of exit. The content is described by PRODEXIT DSECT. Some classes of exits have several subclasses. If in that case IJBVSCL indication is omitted, the exit is defined for all subclasses. Several subclasses can be specified also. If an exit has only one subclass, it is recommended to set the corresponding subclass indication because of future extensions. Subclass is optional for those Classes that allow the subclass specifications.

Note: Classes SVC and FSVC currently do not support single subclasses, therefore any subclass specification is invalid for them and causes Return Code 12.

IJBVTOK the PRODID token previously obtained by PRODID DEFINE. Token is required.

IJBVEXT the exit routine address, the high order bit defines the AMODE on entry of the exit (=0 -> AMODE 24, =1 -> AMODE 31). The exit address is required. AMODE 31 is required if running in ESA mode. If high order bit is not on in VSE/ESA 2.1, DEFINE issues RC 12. The exit routine has to be within the SVA!

IJBVSCP the scope of the exit. Scope is required.

IJBVUSW The 8 byte user word is available for the vendor product, for example, IJBVUSW may hold the pointer to a work area. The contents of IJBVUSW is transferred to the exit. This field is optional and should be cleared to X'00' if no input is available.

IJBVPRIO This one byte field defines the priority of the exit. The priority can be in the range from 0 to 99 (decimal), where 99 is the highest priority and 0 the lowest. The exits are activated in the priority order (high to low), exits with the same priority are activated in FIFO order dependent on the PRODEXIT DEFINE sequence. Exceptions for exit invocations are described in the corresponding exit definition chapters.

IJBVFUPD (in flag IJBVFLAG). Exits defined for update (IJBVFUPD set) are processed prior to all other (monitoring) exits. Within the two groups (update or monitoring) the exits are activated in priority (IJBVPRIO) order.

Note: Proper usage of UPDATE or MONITOR is the vendor's responsibility, VSE does not perform any checks.

IJBVCRIT (in flag IJBVFLAG). An exit defined as critical (IJBVCRIT set) indicates to the system that after the successful processing of that DEFINE service, the system does not ATTACH tasks or ALLOC partitions for which the internal control blocks for the exit activation cannot be obtained. This indication remains valid until all exit are DELETEd that have that flag on. In case of dynamic partitions, allocation is done anyway, but a message is written to the console. In addition, the flag triggers what happens when a second

PSTATE exit for the same task is invoked: if the critical bit is set for the second PSTATE exit, the program is cancelled with error 47 (reason code 3 'second vendor exit invocation invalid'), in the other case the invocation is ignored.

Note: The IJBVCRIT bit is ignored prior to VSE/ESA 2.1.

IJBVDYN (in flag IJBVFLAG). Defines an exit as being eligible for ENABLE/DISABLE,DYN=YES,ID=xxx services to the system. If this bit is on, it triggers an automatic enabling of that exit for SCOPE=SYSTEM. If the scope is not SYSTEM, the exit is only enabled for the dynamic services. The exit stays being enabled for dynamic services until it gets deleted via PRODEXIT DELETE.

Note: This support is currently restricted to exits of the CLASS SVC (ID=SVC) only. RC=12 is returned if the CLASS is not supported.

Note: Any CLASS-specific or SUBCLASS-specific restriction for an exit definition is described in the corresponding exit chapter.

Output: An 8 byte PRODEXIT token passed back in the input PRODEXIT area at label IJBVETK.

Return Codes: passed back in register 15.

0 Request accepted.
8 Authorization problem.
- => PRODID TOKEN index part out of range (>256).
- => Vendor Product Table not defined.
- => Vendor Product Table Entry deleted.
- => Vendor Product TOKEN doesn't match.

12 Control block format error.
- => No CLASS specified.
- => Several CLASSes specified.
- => Specified CLASS not supported.
- => Specified SUBCLASS not supported.
- => PRODEXIT AREA length invalid (<84).
- => SCOPE definition invalid.
- => Priority definition invalid (>99).
- => No EXIT routine address specified.
- => EXIT not in SVA.
- => EXIT routine not AMODE=31 (required in VSE/ESA Version 2).
- => SUBCLASS specification invalid for current CLASS.
- => SCOPE not SYSTEM for CLASS SUP, and SUBCLASS=EXT.
- => SCOPE not SYSTEM for CLASS AIT.
- => IJBVDYN bit on in IJBVFLAG, and CLASS not SVC.

16 System phase $IJBVEND not loaded.
20 GETVIS error, return code from GETVIS in register 0.
24 Request rejected, an exit is already defined for at least one subclass or of the class for the specified TOKEN. The corresponding TOKEN has been returned in the input PRODEXIT area at label IJBVETK.
28 Initial system setup for PRODEXIT services failed due to GETVIS problems.

Cancel Conditions:

X'21' Illegal SVC - unknown function code, invalid invocation of PRODEXIT DEFINE, ASC-mode not PRIMARY or no exits for fast path SVC in CLASS SVC are allowed (for example, SVC 107 and 124).

X'25' Parameter list address is invalid. Results in an addressing exception.

PRODEXIT ENABLE Service

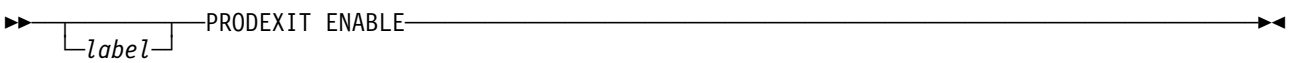
This service enables the exit defined by PRODEXIT DEFINE for exit activation:

System wide if the IJBVSYS flag was set
For current partition if the IJBVPAR flag was set
For current task if the IJBVTSK flag was set

Note: Several ENABLE requests for the same exit are possible.

Requirements For The Caller:

- Authorization: valid PRODEXIT token
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: fast path SVC



Input Parameter Description:

Register 1 and 2 must contain the PRODEXIT token returned by the PRODEXIT DEFINE service.

Output:

Return codes are passed back in register 15.
0 Request accepted.
4 Exit already enabled.
8 Task disabled for PRODEXIT services (GETVIS problem).
16 System phase $IJBVEND not loaded, or no vendor defined in system.

Cancel Conditions:

X'21' Illegal SVC - invalid PRODEXIT token.

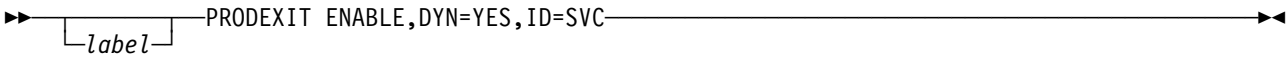
Dynamic PRODEXIT ENABLE

This service enables the exit defined by the coresponding PRODEXIT DEFINE to get activated once for the current task. However, if the current task has also been enabled via the normal PRODEXIT ENABLE service, it stays enabled according to the rules. This service is supposed to be used in a vendor dispatcher exit only.

Note: Several dynamic ENABLE requests for the same exit are possible and can be processed in parallel on different processors in a multiprocessing environment.

Requirements For The Caller:

- Authorization: valid PRODEXIT token and SUPERVISOR STATE
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: BASSM



Input Parameter Description:

Register 1 and 2 must contain the PRODEXIT token returned by the PRODEXIT DEFINE service.

Output:

Return codes are passed back in register 15.

0 Request accepted.
4 Exit not enabled (this should not occur).
8 Task disabled for PRODEXIT services (GETVIS problem).
Dynamic PRODEXIT services not supported for specified ID.
12 Token invalid.
16 System phase $IJBVEND not loaded, or no vendor defined in system.

Cancel Conditions:

X'02' Priviledged operation exception - if not supervisor state.

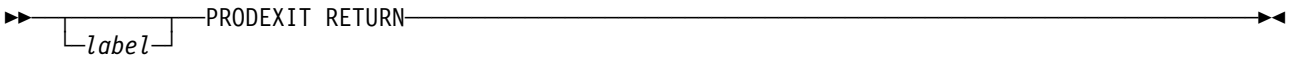
PRODEXIT RETURN Service

The vendor exit has to return with PRODEXIT RETURN. RETURN uses the same PRODEXIT area given to the exit as input, therefore the area must not be changed (except IJBVRC). However, IJBVRC may not be changed if it is a vendor exit without output. Any information returned from the vendor exit is input for the next exit to be called. For example, the vendor exit return code IJBVRC may be analyzed or changed by a following exit. After the return of the last vendor exit (of the subclass) IJBVRC is given to the caller.

Requirements For The Caller:

- Authorization: none
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: SSTATE = BSM, PSTATE = SVC

  Note: Make sure that R14 is not changed during exit processing, it contains the link/return address from the preceding ACTIVATE exit invocation.



Input Parameter Description: none

Output: None.

Cancel Conditions (PSTATE Only):

X'21' ==> Illegal SVC - exit is not active, invalid function code.

ERR2E ==> Deadlock situation during re-occupation of the LTA.

Note: For SSTATE callers that condition leads to unpredictable results.

PRODEXIT DISABLE Service

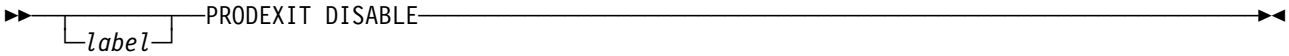
This service disables the exit defined by PRODEXIT DEFINE.

System wide	If the IJBVSYS flag was set
For current partition	if the IJBVPAR flag was set
For current task	if the IJBVTSK flag was set

Note: Several DISABLE requests for the same exit are possible.

Requirements For The Caller:

- Authorization: valid PRODEXIT token
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: fast path SVC



Input Parameter Description:

Register 1 and 2 must contain the PRODEXIT token returned by the PRODEXIT DEFINE service.

Output:

Return codes are passed back in register 15.
0 Request accepted.
4 Exit already disabled.
8 Task disabled for PRODEXIT services (GETVIS problem).
16 System phase $IJBVEND not loaded, or no vendor defined in system.

Cancel Conditions:

X'21' Illegal SVC - invalid PRODEXIT token.

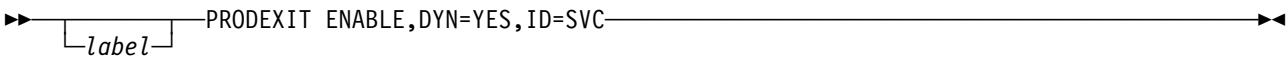
Dynamic PRODEXIT DISABLE

This service disables the exit defined by the corresponding PRODEXIT DEFINE once for the current task. However, if the current task has not been enabled via the normal PRODEXIT ENABLE service, it stays disabled according to the rules. This service is supposed to be used in a vendor dispatcher exit only.

Note: Several dynamic DISABLE requests for the same exit are possible and can be processed in parallel on different processors in a multiprocessing environment.

Requirements For The Caller:

- Authorization: valid PRODEXIT token and SUPERVISOR STATE
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: BASSM



Input Parameter Description:

Register 1 and 2 must contain the PRODEXIT token returned by the PRODEXIT DEFINE service.

Output:

Return codes are passed back in register 15.
0 Request accepted.
4 Exit not enabled (this should not occur).
8 Task disabled for PRODEXIT services (GETVIS problem).
Dynamic PRODEXIT services not supported for specified ID.
12 Token is invalid.
16 System phase $IJBVEND not loaded, or no vendor defined in system.

Cancel Conditions:

X'02' Priviledged operation exception - if not supervisor state.

PRODEXIT DELETE Service

The exit is disabled (if not yet done) and the exit definition is deleted. PRODEXIT DELETE waits for PRODEXIT RETURN, if the exit is active.

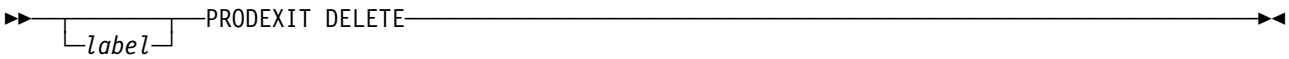
Vendor exits **must not** issue PRODEXIT DELETE.

Note: When a vendor product is deleted (using PRODID DELETE) from the system, all associated vendor exits (defined by PRODEXIT DEFINE) are deleted, too. If one of these exits is active, PRODID DELETE waits for the PRODEXIT RETURN from these exits.

Vendor exits that were defined using one DEFINE can only be deleted together.

Requirements For The Caller:

- Authorization: valid PRODEXIT token
- AMODE: ANY
- RMODE: ANY
- ASC-mode: Primary
- Invocation: SVC



Input Parameter Description:

Register 1 and 2 must contain the PRODEXIT token returned by the PRODEXIT DEFINE service.

Output:

Return codes are passed back in register 15.
0 Request accepted.
16 System phase $IJBVEND not loaded.

Cancel Conditions:

X'21' Illegal SVC - invalid PRODEXIT token, may be exit already deleted, or invalid invocation of PRODEXIT DELETE.

PRODEXIT DSECT Service

This service generates one DSECT that describes the in/output for the following services:

- The input and output expected by PRODEXIT DEFINE.
- The input and output expected by PRODEXIT ACTIVATE.
- The input and output expected by PRODEXIT CHECK.
- The output expected by PRODEXIT RETURN.

PRODEXIT DSECT describes the common part, which is input for the PRODEXIT services and the vendor exits, as well as the output area for information returned to the PRODEXIT issuer. The PRODEXIT DSECT holds an address (at label IJBVINF) that points to the PRODEXIT extension of the corresponding subclass at exit entry. This extension is used to communicate between the exit and the VSE component that activated the exit. The unique subclass extension part labels start with **IJBVx**, where **x** is defined as

B Class = BAM
C Class = SVC
F Class = FSVC (fast SVC)
L Class = LNG (languages)
P Class = PSUP (supervisor PSTATE)
S Class = SUP

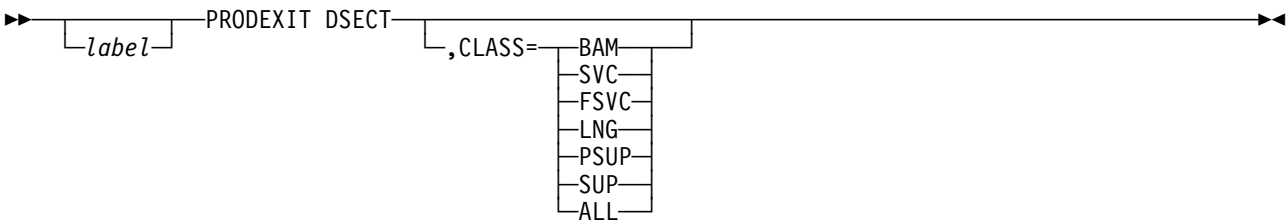
The input/output for the classes AIT, DOC, and DOCP is not described by the PRODEXIT DSECT but by the following macros:

MAPARCMD for class = AIT (see “Vendor Exit - VSE/AF Attention Routine” on page 52)
IJBCSMX for class = DOC (see “Message Processing Exit” on page 48)
IJBCNMEX for class = DOCP

The first eight bytes of the extension are common to all exits.

Area Location:

- RMODE: ANY, in private or shared area dependent on PRODEXIT service.
- ASC-mode: Primary



CLASS=ALL generates the labels for all CLASSES and suppresses double definitions if more than one CLASS is defined in the same assembly.

Vendor Exits

The vendor exits must adhere to the following specifications:

- To use the vendor exits, the application must be authorized. The valid token is obtained using the PRODID macro, and the user must be authorized (DTSECTAB) to access IJBVEND and the product's exit routine.
- An exit can be dynamically enabled or disabled without affecting other exits for the same event type.
- The various exit specifications are stacked per event and are invoked as needed for one specific exit.

The following vendor exits are available:

Vendor Exit - VSE/AF Supervisor (SUP): The following command generates the product extension area (also called DSECT) for this class:

```
name PRODEXIT DSECT, CLASS=SUP
```

The following exits are supported:

- Post SIO/SSCH
- I/O interrupt
- Program check interrupt
- External interrupt
- Exchange phase
- Program retrieval (pre and post fetch)

Post SIO/SSCH:

Class SUP

Subclass POSTSIO

Purpose Monitoring

Location Before current SGVSEPT probe points in SGSCHED

Exit Type SSTATE

RID SYSTEMID (=0, no page faults allowed)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- The CUU that the I/O was issued to
- The CCB address

Output None

I/O Interrupt:

Class SUP

Subclass IOINT

Purpose Monitoring

Location Before current SGVSEPT probe point

Exit Type SSTATE

RID SYSTEMID (=0, no page faults allowed)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- CSW
- CUU of device causing the interrupt

Output None

Program Check:

Class SUP

Subclass PCK

Purpose

- Monitoring
- Error recovery

Note: PCK exits are not invoked during processing of SVC exits.

Location Before current SGVSEPT probe point in SGPCK

Exit Type SSTATE

RID SYSTEMID (=0, no page faults allowed)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Program check old PSW
- Program interruption code (loc X'8C'-X'8F')
- RID at interruption time
- TID and PIK of interrupted task
- Page fault address (location X'90') if page fault
- Exception access identification (location X'A0') if page fault due to access to data space
- General purpose registers and access registers

Output A return code is set up in IJBVRC:

00 Continue normal program check handling

04 Only for update exits (IJBVUPD). Continue with PSW, general purpose/access registers as specified in communication area.

Note: The first update exit that returns a RC=4 causes a skip of all following update exit invocations. Only monitor type exits will then get control prior to the execution of the RC=4 processing. If another update exit follows, a bit (IJBVSUPD in byte IJBVFLAG) is set in the PRODEXIT area indicating that an exit was skipped.

08 Only for update exits (IJBVUPD). Ignore program check and return to interrupted task.

External Interrupt:

Note: The only predictable exit invocation for SUBCLASS=EXT is provided for SCOPE=SYSTEM. Therefore any other SCOPE is rejected during PRODEXIT DEFINE processing (RC=12).

Class SUP

Subclass EXT

Purpose

- Control APPC/IUCV/VMCF
- Monitoring

Location Before current SGVSEPT probe point in SGNUC (after ENTEXT)

Exit Type SSTATE

RID SYSTEMID (=0, no page faults allowed)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- External old PSW
- External interruption code (loc X'84'-X'87')
- TID and PIK of interrupted task
- TID of task whose timer expired (if any)
- RID of interrupted task
- Service signal interrupt parameter word (EXTPARWD)

Output A return code is set up in IJBVRC:

00 Continue external interrupt processing.
08 Only for update exits (IJBVUPD). Ignore external interrupt.

Program Retrieval - Exchange Phase:

Class SUP

Subclass EXPHASE

Purpose

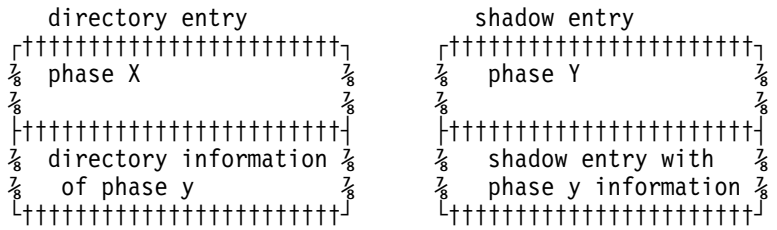
- Security
- Monitoring
- Gaining a probe point

Location The exit is called for:

- SVC 1 (FETCH), SVC 2, SVC 4 (LOAD), SVC 23, SVC 51, SVC 65 (CDLOAD)

Note, that the phase name in the user area is never exchanged.

- CDLOAD/CDDELETE Processing: The exit is placed at the beginning of the CDLOAD/CDDELETE processing, before the CDLOAD anchor table is searched for the requested phase. This allows an exchange of the phase name. If the phase name is exchanged, the follow-on processing is done with this exchanged phase name. The user-provided phase name is no longer of interest. The entry in the anchor table contains the exchanged phase name.
- SVC 2 Move Mode Processing The exit is called before the SVA is searched for the requested phase. In case the phase is not found in the SVA, the common processing is called (with the original phase name). Note, that during this processing the exit is called again.
- Common Processing (called by SVC 1, SVC 4, SVC 23, SVC 51, SVC 65, and SVC 2 if phase is not in the SVA). The exit is placed at the beginning of the common processing to allow an exchange of the phase name. From now on all processing is done with the changed phase name. The user provided-phase name is no longer of interest. In case the user has provided a directory entry, both the directory entry and the shadow entry are filled with phase information of the exchanged phase name. The phase name in the user-provided directory is not exchanged.



The directory entry contains a pointer to the shadow entry. If the DE is used for follow-on requests, it is checked whether the directory entry and the shadow entry describe the same phase. To avoid problems with the different phase names, the exit is called and must exchange the phase name. Otherwise the information in the directory entry is not considered and the original phase x is loaded.

Exit Type SSTATE

RID REENTRID (reentrant programming required)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- PIK of the current task (2 bytes)
- task-id (TID) of the current task (2 bytes)
- SVC code that generated the request (1 byte)
- name of program to be fetched (8 bytes)

Output The output (if any) is returned in the input area (pointer to area in field IJBVINF).

A return code is set up in IJBVRC:

0 Continue processing.
4 Only for update exits (IJBVUPD). Change of phase name requested. The phase name is changed to the phase name supplied in the input area. The new phase name is in IJBVSFEP (8 bytes - if IJBVRC = 4).
8 IGNORE Skip FETCH / LOAD / CDLOAD / CDDELETE processing.

Note: When a vendor exit requests to exchange the phase name, it is in the responsibility of the vendor product to guarantee the availability of the exchanged phase. If the phase cannot be loaded, results may be unpredictable.

Program Retrieval - Pre Fetch:

Class SUP

Subclass PREFCH

Purpose

- Security
- Monitoring
- Gaining a probe point

Location The exit is called after directory task process (directory search) and runs as service owner (user task). It is only called if the directory search ended with return code 0. It is not called if the phase is in the SVA.

Exit Type SSTATE

RID REENTRID (reentrant programming required).

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- PIK of the current task (2 bytes)
- Task ID (TID) of the current task (2 bytes)
- SVC code that generated the request (1 byte)
- Name of program to be fetched (8 bytes)
- Library name of program (7 bytes)
- Sublibrary name of program (8 bytes)
- Directory entry (DE=VSE format - 40 bytes)

Output A return code is set up in IJBVRC:

0 Continue processing.
4 Only for update exits (IJBVUPD). Reject load request (security violation is posted to the user).

Program Retrieval - Post Fetch:

Class SUP

Subclass POSTFCH

Purpose

- Security
- Monitoring

Location The exit is placed:

- After the program fetch task processing (loading the phase) and runs as service owner (user task). It is called even if an error occurred.
- After moving a move mode phase to the LTA during SVC 2 processing.

Exit Type SSTATE

RID REENTRID (reentrant programming required).

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- PIK of the current task (2 bytes)
- Task ID (TID) of the current task (2 bytes)
- SVC code that generated the request (1 byte)
- Name of program to be fetched (8 bytes)
- Library name of program (7 bytes)
- Sublibrary name of program (8 bytes)
- Return code
- Directory entry (DE=VSE format - 40 bytes)

Output None.

Vendor Exit - VSE/AF Supervisor (PSUP): The following command generates the product extension area (also called DSECT) for this class:

```
name PRODEXIT DSECT, CLASS=PSUP
```

The following exit is supported:

- End-of-task ($IJBSEOT)

End-Of-Task - $IJBSEOT Phase:

Class PSUP

Subclass EOT

Purpose The exit is implemented to allow task clean-up.

Location The exit is placed prior to FREEVIS ALL into the $IJBSEOT routine and is called for main- and subtasks in the AMODE defined by the exit address.

Exit Type PSTATE

RID USERTID

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- PIK of current task (2 byte)
- Task ID (TID) of current task (2 byte)
- First cancel code (TIBCNCL) (1 byte)
- Second cancel code (TIBCNCL2) (1 byte)
- Flag byte to indicate main- or subtask termination (1 byte)

Output None.

Vendor Exit - VSE/AF Supervisor Call: The vendor exit gets control whenever a SVC with exception of SVC 107 (fast path SVC) or SVC 124 (Vendor SVC) is issued.

The following command generates the product extension area (also called DSECT) for this class:

```
name PRODEXIT DSECT, CLASS=SVC
```

Note: Subclass SVC code is subject for future extension. Any subclass specification during PRODEXIT DEFINE is rejected with RC=12!

Class SVC

Subclass none

Purpose

- SVC screening
- Monitoring
- Adding vendor SVC numbers
- Ignoring SVCs
- Expansion on IBM SVC processing
- Security

Location Before current SGVSEPT probe point in SGNUC (after ENTSVC).

Exit Type SSTATE

RID REENTRID

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- SVC old PSW

- SVC interruption code
- TID and PIK of interrupted task
- Register and access register at time of interruption

Output A return code is set up in IJBVRC:

00 Continue normal SVC handling.

04 Only for update exits (IJBVUPD).Replace SVC OLDPSW by IJBVCPSW. Substitute SVC number and continue with general purpose/access registers as specified in the communication area.

08 Only for update exits (IJBVUPD). Ignore SVC and go to routine where modified SVC old PSW address is pointing to (IJBVCPSW) using general purpose/access registers as specified in the communication area. (Changing the registers is possible for VSE/ESA releases starting with 2.1.1, or with APAR DY43684.)

Note: The first update exit that returns a RC=8 causes a skip of all following update exit invocations. Only monitor type exits will then get control prior to the execution of the RC=8 processing. If another update exit follows, a bit (IJBVSUPD in byte IJBVFLAG) is set in the PRODEXIT area indicating that an exit was skipped.

12 Only for update exits (IJBVUPD). Cancel user due to security violation. Only allowed when SVC issuer (user) had RID=USERTID. In case of security violation, the vendor exit can pass additional information in the PRODEXIT extension area, field IJBVCVIN. This information appears in the cancel message.

Vendor Exit - VSE/AF Fast Path Supervisor Call: The vendor exit gets control whenever a SVC 107 (fast path SVC) is issued.

The following command generates the product extension area (also called DSECT) for this class:

```
name PRODEXIT DSECT, CLASS=FSVC
```

Note: Subclass FSV code is subject for future extension. Any subclass specification during PRODEXIT DEFINE is rejected with RC=12!

Class FSVC

Subclass none

Purpose

- SVC screening
- Monitoring
- Security

Location New probe point in SGNUC, after SVC 107 path is entered.

Exit Type SSTATE

RID SYSTEMID (=0, no page faults allowed)

Communication area SVA, RMODE=ANY, shared area

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- SVC old PSW

- SVC interruption code
- RID of interrupted task
- TID and PIK of interrupted task
- Register and access register at time of interruption

Output A return code is set up in IJBVRC:

00 Continue normal SVC handling.
12 Only for update exits (IJBVUPD). Cancel user due to security violation. Only allowed when SVC issuer (user) had RID=USERTID. Hard wait otherwise. In case of security violation, the vendor exit can pass additional information in the PRODEXIT extension area, field IJBVFVIN. This information appears in the cancel message.

Vendor Exit - VSE/AF Basic Access Method: The following command generates the product extension area (also called DSECT) for this class:

```
name PRODEXIT DSECT, CLASS=BAM
```

The following exits are supported:

- Pre-OPEN
- Pre-CLOSE
- EOX processing for sequential disk
- Common VTOC handler
- Post-OPEN
- Post-CLOSE

Invocation: When exits are invoked in a B-transient phase (Post-OPEN, Post-CLOSE, Pre-OPEN, and Pre-CLOSE), the logical transient area is freed before the exit is taken.

Pre OPEN:

Class BAM

Subclass PREOPEN

Purpose

- Monitoring
- Ignoring OPEN
- Replacing DTFs and ACBs

Location At the beginning of OPEN, just before the system does any action with the DTF/ACB ($$BOPEN1).

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Pointer to the DTF/ACB (4 bytes), IJBVBDTF
- Flag byte IJBVBFLG

– X'80', IJBVBJCT, indicates request is from JCL

Output A return code is set up in IJBVRC:

00 Continue with normal OPEN processing for this DTF/ACB.
04 Ignore OPEN of this DTF/ACB. No system action is done with this DTF/ACB.
12 Change of DTF/ACB address requested. The DTF/ACB address in users parameter list is replaced by the address returned by the exit in IJBVBDTF and OPEN continues with this DTF/ACB.

Note: If a vendor exit passes an invalid return code at return, the system continues normal processing.

Pre CLOSE:

Class BAM

Subclass PRECLOSE

Purpose

- Monitoring
- Ignoring OPEN
- Replacing DTFs and ACBs

Location At the beginning of CLOSE, just before the system does any action with the DTF/ACB ($$BCLOSE).

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Pointer to the DTF/ACB (4 bytes), IJBVBDTF

Output A return code is set up in IJBVRC:

00 Continue with normal CLOSE processing for this DTF/ACB.
04 Ignore CLOSE of this DTF/ACB.
08 Perform **no** rewind, only for DTFMT and DTFPH for tape.
12 Change of DTF/ACB address requested. The DTF/ACB address in users parameter list is replaced by the address returned by the exit in IJBVBDTF and CLOSE continues with this DTF/ACB.

Note: If a vendor exit passes an invalid return code at return, the system continues normal processing.

EOX Processing:

Class BAM

Subclass BAMEOX

Purpose Monitoring

Location After EOX is recognized for sequential disk and before the next EXTENT if any is obtained. The exit is **not** invoked for end of extent during CLOSE.

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Pointer to the DTF (4 bytes), IJBVBDTF

Output none

CVH Processing:

Class BAM

Subclass BAMCVH

Purpose

- Monitoring
- Ignoring OPEN

Location At the very beginning of the common VTOC handler (IJJHCVH0).

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Address of CVH parameter list IJJHCPL (4 bytes), IJBVBCPL

Output A return codes is set up in IJBVRC:

00 Continue with normal Common VTOC Handler processing.
04 Return immediately to caller. In this case it is the vendors responsibility to maintain the Common VTOC Handler interface to the caller. Return code 4 is not allowed for OPEN and CLOSE VTOC requests.

Note: If a vendor exit passes an invalid return code at return, the system continues normal processing.

Post OPEN:

Class BAM

Subclass POSTOPEN

Purpose

- Monitoring
- Replacing DTFs and ACBs

Location At the end of OPEN process, before the next DTF is handled or before return to user using SVC 11 if the last or only DTF has been handled ($$BOPEN1).

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Pointer to the DTF/ACB (4 bytes), IJBVBDTF
- Flag byte IJBVBFLG
  - X'80', IJBVBJCT, indicates request is from JCL

Output A return codes is set up in IJBVRC:

00 Continue with normal OPEN processing.
12 Change of DTF/ACB address requested. The DTF/ACB address in users parameter list is replaced by the address returned by the exit in IJBVBDTF and normal processing continues.

Notes:

1. If a vendor exit passes an invalid return code at return, the system continues normal processing.
2. Post OPEN exit invocation is only guaranteed for DTFMT, DTFPH for tape, and DTFs for disk. The Invocation for other DTFs is unpredictable.

Post CLOSE:

Class BAM

Subclass POSTCLOS

Purpose

- Monitoring
- Replacing DTFs and ACBs

Location At the end of CLOSE process, before the next DTF is handled, or before return to user using SVC 11 if the last or only DTF has been handled ($$BCLOSE).

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Pointer to the DTF/ACB (4 bytes), IJBVBDTF

Output A return codes is set up in IJBVRC:

00 Continue with normal CLOSE processing.
12 Change of DTF/ACB address requested. The DTF/ACB address in users parameter list is replaced by the address returned by the exit in IJBVBDTF and normal processing continues.

Notes:

1. If a vendor exit passes an invalid return code at return, the system continues normal processing.
2. Post CLOSE exit invocation is only guaranteed for DTFMT, DTFPH for tape, and DTFs for disk. The Invocation for other DTFs is unpredictable.

Vendor Exit - VSE/AF Console Support

Message Processing Exit:

Class DOC

Subclass MSG

Purpose

- Monitoring every message (using WTO/WTOR or SVC 0/15)
- Updating control information and text
- Suppressing or automatically replying to messages

Location The exit is invoked after converting the message to a common format, including the message prefix, and before queueing the message for delivery. If OCCF message processing is active, it is also invoked before the exit.

Exit Type SSTATE

RID SYSTEMID (no page faults allowed, emercency messages only) or REENTRID (page faults allowed)

Communication area SVA, RMODE=ANY, partition GETVIS

Input IJBVINF points to the communication area (input and output area). The area holds the following information at exit entry:

- Message parameters that cannot be changed (like originating jobname).
- The adress of a message area, also described by IJBCSMX and containing data may be updated:
  - Name of the target console
  - Routing and descriptor codes
  - Presentation attributes
  - Message text lines (space for up to 12 lines is provided)

  This area is initially filled with original message data.
- The address of an automatic reply area, consisting of a 2-byte length followed by up to 126 bytes of reply text. This area is initially empty (length field is zero). An automatic reply must be prepared in this area, as if it was entered from a console (but without the leading reply-id). An automatic reply may be provided also when no READ was yet issued. In such a case, the reply is processed automatically for a following stand-alone read , a following stand-alone WTOR (that is a WTOR with a 'just-one-blank' text) or discarded otherwise.
- A flag byte for requesting one of the following processing options:
  - Suppress the message.
  - Reply to message automatically.
  - Route message to a console with AUTO attribute
  - Update message control information
  - Update message text

Output None.

Note:

1. This exit is not intended for intercepting/redirecting messages through an alternate queueing mechanism.
2. The exit is not invoked for DOM requests.
3. In some cases, for example, when no queue space is immediately available for an extended message, the exit may be invoked several times for the same message, with identical message contents.

Command/Reply Processing Exit: This exit allows to monitor every input, entered via MGCRE or SVC 30, and to reject it or to update input text.

Class DOC

Subclass CMDREP

Purpose

- Monitoring every input (entered using MGCRE or SVC 30)
- Rejecting input
- Updating input text

Location The exit is invoked after converting console input to a common format, and before processing or queueing it for delivery. If OCCF input processing is active, it is also invoked before the exit.

Exit Type SSTATE

RID REENTRID (page faults are allowed).

Communication area Partition GETVIS, RMODE=ANY

Input IJBVINF points to the communication area (input and output area), which is described by the IJBCSMX mapping macro. The area holds the following information at exit entry:

- Input parameters that cannot be changed (like originating console name).
- The address of an input area, consisting of a 2-byte length followed by up to 126 bytes of input text. This area is initialized with the original input.
- A flag byte for requesting one of the following processing options:
  - Reject the input
  - Update input text

Output None

Notes:

1. This exit is neither invoked for automatic replies, generated by a message processing exit (see “Message Processing Exit” on page 48), nor for EXPLAIN requests.
2. Any return code setting in IJBVRC is ignored.

Full Scale input processing exit

This exit allows to monitor every console input entered on the command line“, every timer interrupt, reconnect, message pending, action message received, alert posted, suspended, and unrecoverable error.

Class DOCP

Subclass FINPUT

Purpose

- Monitor command inputs.
- Update these inputs.
- Monitor timer interrupts.
- Monitor reconnect.
- Monitor message pending.
- Monitor action/decision message received.

- Monitor alert posted.
- Monitor console suspended.
- Monitor unrecoverable error.

Location The exit is invoked for every user input in console mode and for other events, as indicated in field CEXEVNT. The exit has the option to take over control of console output ('vendor mode'). In this mode, the screen is fully managed by exit code, while input is still handled by standard code and passed to the exit, as in console mode.

Exit Type PSTATE

Input IJBVINF points to the communication area(input and output area), which is described by the IJBCNMEX mapping macro. The area holds the following information at exit entry:

- Addresses of screen and message definitions. Mappings defined in IJBDEF macro.
  - address of panel data table
  - address of PF-key data table
  - address of message data table
- Terminal characteristics
  - number of screen lines
  - number of screen columns
  - DBCS capability: 'Y' or 'N'
  - ext data stream capability: 'Y' or 'N'
  - number of supported colors
  - supported color attributes
  - number of supported highlites
  - supported highlite attributes
- Event flags. Flags CEXEMSGP, CEXEALRT and CEXESUSP, once set, remain on until the exit routine leaves vendor mode. These flags may therefore be set in combination with other flags, which are mutually exclusive.
  - user input
  - timer interput
  - action/decision message received
  - reconnect (after PF3/PF4)
  - message pending
  - alert posted
  - console suspended
  - unrecoverable error

Output

- Processing flags, timer interval
- The options'use updated input' and 'enter vendor mode' are only processed for 'user input' events.

- Since the screen is controlled in vendor mode by the exit, the exit is also responsible for issuing a write, that unlocks the keyboard, for every user input passed to it.
- when entering vendor mode, any pending timer interupt is cancelled.
- The option 'set timer' may only be specified in conjunction with 'enter vendor mode' or while vendor mode is active. When specified in vendor mode, any already scheduled interrupt is cancelled. A new interrupt is set only if the value passed in CEXPTIMS is > 0.

Vendor Exit - VSE/AF Attention Routine: The Attention Routine supports two exits that allow to customize existing system commands or to add new commands. They are not intended for modifying command input (the CMDREP exit described in “Command/Reply Processing Exit” on page 50 should be used for this purpose).

Note: The only predictable exit invocation for CLASS=AIT is provided for SCOPE=SYSTEM. Therefore SCOPE=PARTITION/TASK is rejected during PRODEXIT DEFINE processing (RC=12).

Pre-Scan Exit:

Class AIT

Subclass CMD1 (Pre-Scan)

Purpose This exit allows to intercept all VSE/ESA system commands that are processed by or routed through the Attention Routine, and to provide customized versions of such commands.

Location The exit is invoked before AR parses the command.

Exit Type PSTATE

RID USERTID

Communication area SVA, RMODE=ANY

Input The pointer for this vendor exit is different from the others, as described in the following paragraph.

IJBVINF points to an area containing the command string, along with other command attributes like its origin, and described by the mapping macro MAPARCMD.

Output A return code is set up in IJBVRC:

0 Command was taken over by the exit.

Note: The first update exit that returns a RC=0 causes a skip of all following update exit invocations. Only monitor type exits will then get control prior to the AR CMD processing. If another update exit follows, a bit (IJBVSUPD in byte IJBVFLAG) is set in the PRODEXIT area indicating that an exit was skipped.

4 Command was not recognized.

Post-Scan Exit:

Class AIT

Subclass CMD2 (Post-Scan)

Purpose This exit allows to define and support new commands that are routed through the Attention Routine, but are not recognized as known VSE/ESA system commands.

Location The exit is invoked when the command is not recognized as an AR or active subsystem command.

Exit Type PSTATE

RID USERTID

Communication area SVA, RMODE=ANY

Input The pointer for this vendor exit is different from the others, as described in the following paragraph.

IJBVINF points to an area containing the command string, along with other command attributes like its origin, and described by the mapping macro MAPARCMD.

Output A return code is set up in IJBVRC:

0 Command was taken over by the exit.
4 Command not recognized.

Vendor Exit - VSE/ESA Language Environment:

"How to Use this Exit" on page 57 describes the various approaches to using this exit.

Class LNG

Subclass LNGOPEN

Purpose The exit allows a disk or tape manager to intercept file open processing for LE/VSE-enabled languages, and allows it to provide some information or cause it to bypass some of the pre-open checks. These languages perform some pre-open checking to enable them to return correct statuses to the programs. When a disk or tape manager provides some information, such as logical unit, during open, the language checks will fail, and the open is not issued. This exit allows this information to be provided to the language, or to request the language to bypass these checks. If these checks are bypassed, some of the statuses returned may be incorrect.

Location The exit is invoked by LE/VSE when the LE/VSE Message file is opened, or when an open is requested in an LE/VSE-enabled language (COBOL/VSE or PL/I VSE). The exit is invoked before building the DTFSD, DTFDA, DTFDU, DTFMT, DTFPR, or DTFCD, and prior to opening the file. It is also invoked by the COBOL/VSE compiler prior to checking the logical unit for the compiler work files.

Notes:

1. If errors are detected after the exit is invoked, the OPEN is not issued.
2. The exit is not invoked for files that are accessed using an ACB.

Exit Type PSTATE

RID USERTID

Communication area Partition GETVIS, RMODE=ANY

Input and Output IJBVINF points to the communication area (input and output area). The area holds the information shown in the following table at exit entry.

The Type column contains “Input” if it is provided by the language, “Output” if it is provided by the exit, or “Input/Output” if it is provided by the language but may be modified by the exit.

Field	Size	Type	Description
IJBVLENV	H	Input	Length of area
IJBVPIK	H	Updated by supervisor	PIK of current task
IJBVTIK	H	Updated by supervisor	TIK of current task
Reserved	H		Reserved
IJBVLLVL	F	Input	Level number of the parameter list.
IJBVLANG	CL8	Input	The Language product that is invoking the exit, for example, COBOL, PL/I, LE.
IJBVLNAM	CL8	Input	The name from the COBOL ASSIGN statement, or PL/I File DECLARE statement.
IJBVLLUN	H	Input	The logical unit number from the COBOL ASSIGN statement, or PL/I File DECLARE statement, or zero if not provided.

IJBVLOPM	X	Input	Open mode with values as follows:
			`IJBVLIN  X'01' - Input`
			`IJBVLIO  X'02' - I/O (for example, TYPEFLE=INPUT and UPDATE=YES)`
			`IJBVLOUT X'03' - Output`
			`IJBVLEXT X'04' - Extend (append)`
			`IJBVLWRK X'05' - Work (for example, compiler work file)`

IJBVLDEV	X	Input/ Output	Device type for file with values as follows:
			`IJBVLDSD X'01' - SAM disk`
			`IJBVLDDA X'02' - DAM disk`
			`IJBVLDVS X'03' - VSAM disk`
			`IJBVLDUT X'04' - Unlabeled tape`
			`IJBVLDLT X'05' - Labeled tape`
			`IJBVLDCD X'06' - Card`
			`IJBVLDPR X'07' - Printer`
			`IJBVLDUA X'08' - Unassigned (UA)`
			`IJBVLDIG X'09' - Assigned to IGNORE`
			This field can be modified by the exit to change the device type of the file. For instance, a labeled tape can be changed to an an unlabeled tape or a disk file can be changed to a tape file. This field should not be modified for DAM files or files with an open mode (IJBVLOPM) of WORK (IJBVLWRK).
			Note: The device type is set to SAM, DAM, or VSAM-managed SAM when a DLBL is present. It is set to labeled tape when a TLBL is present. It is set to one of the other device codes when neither a DLBL or TLBL is present, according to the type of device to which the logical unit number (IJBVLLUN) is assigned.

Field	Size	Type	Description
IJBVLLBA	A	Input/ Output	The address of the DLBL or TLBL retrieved by the language product. This is present for Disk, or labeled tape files. The fields within the DLBL or TLBL can be altered as required. The length of the DLBL or TLBL (IJBVLLBL) can be increased up to length of the buffer containing the DLBL or TLBL (IJBVLLBB). The address can be changed to the address of an area acquired by the exit, and the length (IJBVLLBL) updated to reflect the new length. The fields that are currently used from the DLBL are as follows: • File ID (if not provided in field IJBVLFID). • Logical unit number (for SAM DLBL if not provided in field IJBVLLUO) • Blocksize (for SAM DLBL if not provided in field IJBVLBSO) • Catalog name (for VSAM DLBL) The fields that are currently used from the TLBL are as follows: • File sequence number (for multi-file tape).
IJBVLLBL	F	Input/ Output	The length of the DLBL or TLBL retrieved by the language product using the `LABEL FUNCT=GETLBL` macro.
IJBVLLBB	F	Input	The length of the buffer containing the DLBL or TLBL retrieved by the language product using the `LABEL FUNCT=GETLBL` macro.

IJBVLOPT	XL2	Input/ Output	Other open options.
			
			`IJBVLORV x'80' - OPEN REVERSED specified`
			`IJBVLNRW x'40' - OPEN NO REWIND specified`
			`IJBVLASC x'20' - ASCII tape file`
			`IJBVLOPF x'10' - OPTIONAL file (COBOL only)`
			`IJBVLCBM x'08' - MODE=C (column binary mode)`
			`(3505 and 3525 card devices only)`
			`IJBVLOMR x'04' - MODE=O (Optical mark read)`
			`(3505 card devices only)`
			`IJBVLRCE x'02' - MODE=R (Read Column Eliminate)`
			`(3505 and 3525 card devices only)`

IJBVLPUB	X	Input/ Output	Pub code This field is provided by the language product for unlabeled tape, card and printer devices, and may be modified by the exit. It is used to determine the device type when a DTF is built.
IJBVLFEX	X	Output	File exists flag set by exit to X'01' if file currently exists, or X'02' if file does not exist. If it is not set (left as X'00'), and the file is being opened for input or I/O, the language attempts to determine if the file exists.
IJBVLLUO	H	Output	The logical unit number determined by the exit. If it is provided, the subsequent processing by the language uses this logical unit number. If it is not provided for a SAM file, the language attempts to determine the logical unit from the DLBL or COBOL ASSIGN statement or PL/I File DECLARE statement.
IJBVLFID	CL44	Output	This is the file ID for the file, with any vendor-supplied tape or disk manager control characters removed. If it is not provided, the language uses the file-id from the DLBL or TLBL statement for a SAM file.

Field	Size	Type	Description
IJBVLLEX	H	Output	Extent number of the last volume. This is applicable to SAM files only. If it is not set (left as zero), the language attempts to determine the extent number of the last volume from the DLBL/EXTENT statements. The extent number is used by COBOL when the CLOSE REEL/UNIT statement is used. **Note:** If the vendor-supplied tape or disk manager provides extents as required, this field can be set to -1. This disables the CLOSE REEL/UNIT statement for COBOL.

IJBVLRFI	H	Input	Record format from program (zero if not provided) as follows:
			`IJBVLRFF x'80' - Fixed`
			`IJBVLRFV x'40' - Variable`
			`IJBVLRFU x'20' - Undefined`
			`IJBVLRFB x'10' - Blocked`
			`IJBVJRFS x'08' - Spanned`
			`IJBVLRFA x'04' - ASA control characters`
			`IJBVLRFM x'02' - Machine control characters`
			For example, x'90' for Fixed Blocked.

IJBVLRSI	F	Input	Record length from program (zero if not provided)
IJBVLBSI	F	Input	Block size from program (zero if not provided)
IJBVLRFO	H	Output	The record format for the existing file if available, with the same flags as IJBVLRFI. If it is not set (left as zero), the file is being opened for input or I/O, and it is a VSAM-managed SAM file, the language determines the record format from the VSAM catalog. For a file opened for Input, I/O, or Extend, the record format from the program is checked to ensure it matches the record format for the input file.
IJBVLRSO	F	Output	The record length of the existing file if available. If it is not set (left as zero), and the file is being opened for input, I/O, or EXTEND, and it is a VSAM-managed SAM file, the language determines the record length from the VSAM catalog. For a file opened for Input, I/O, or Extend, the record length from the program is checked to ensure it matches the record length for the input file. If the record length is not specified in the program (for example, RECORD CONTAINS 0 CHARACTERS for COBOL), the record length provided in this field is used (if non-zero).
IJBVLBSO	F	Output	The block size of the existing file or output file. If it is not set (left as zero), the language attempts to determine the block size. For a VSAM-managed SAM file, the VSAM catalog is checked. For a SAM file, the block size from the DLBL is used if present. This block size overrides the block size specified within the program for a disk or tape file with the record format specified as blocked. If the record format is fixed, the block size must be a several of the record size.
IJBVLLMA	A	Output	The address of the LIOCS logic module to be placed in the DTF after open. This may be used to replace the IBM-supplied logic module.
IJBVLLMS	A	Output	The address of a fullword to return the address of the logic module in the DTF prior to being overwritten by the logic module address supplied above.

A return code is set up in IJBVRC:

00 Continue processing.

04 Continue processing, but bypass any pre-open checks. The open processing continues if the following errors are detected.

- No EXTENT card has been provided for a SAM DLBL, and the logical unit number was not provided on the COBOL ASSIGN statement or PL/I File DECLARE statement.
- The logical unit is not assigned.
- The file has been opened for Input or I/O, and the file does not exist.

Other Don't continue processing, and fail the open.

How to Use this Exit: Depending on how much effort a Vendor may wish to expend, and the capabilities of the Vendor product, the exit can do one of the following:

Approach 1 - Minimal Approach

The minimal approach would be for the Vendor Exit to set IJBVRC to 4. This causes the languages to skip the checking of the device. This would allow the files to open, but the restrictions that are currently present when the Vendor products are present would still apply.

Some of the known restrictions are as follows:

1. Support for OPTIONAL files in COBOL is not provided.
2. Some file statuses under COBOL are not returned correctly.
3. Support for CLOSE REEL/UNIT in COBOL is not provided.

Approach 2

The above restrictions are caused when the language product cannot determine whether the file exists, or cannot determine the file attributes. The language product cannot perform these checks because the file-id on the DLBL or TLBL might not be the actual file-id, or because the logical unit number is not available.

The exit could perform the updating of the DLBL or TLBL statement including the assignment of the logical unit during the exit processing instead of during the OPEN. If the file-id on the DLBL or TLBL statement is not the actual file-id, the actual file-id should be returned.

This would involve setting the following fields:

- Logical unit number
- File-id (if the file-id on the DLBL or TLBL does not match the actual file-id)
- File exists flag (Input and I/O only)
- Number of volumes

Approach 3

In addition to the processing for the previous approach, if the vendor product has information about the file such as record format, record length, and block size that is not available directly from the VSE/ESA operating system, the record format, record length, and block size can also be returned.

COBOL has the ability to enable a program to read a file with fixed length records of any length (RECORD CONTAINS 0 CHARACTERS), or with any block size (BLOCK CONTAINS 0 RECORDS). If this information is returned by the exit, it enables a COBOL program to read these files. This information is currently retrieved from the VSAM catalog by COBOL for VSAM-managed SAM files.

PL/I has the ability to enable a program to read a file with any record format, record length, and block size. If this information can be returned by the exit, PL/I could then open the file without requiring the program to specify this information.

This approach provides additional capabilities to the COBOL and PL/I languages when the vendor-supplied tape or disk manager is operating and able to provide the additional information. This brings the VSE versions of COBOL and PL/I closer to the capabilities provided by the MVS Operating system.

Part 3. Documentation and National Language Support

Chapter 4. VSE Customer Documentation

Certainly, little causes customer dissatisfaction with products as much as poorly written technical publications do, publications that prevent a proper understanding of the product's functions. VSE documentation looks at the following:

- The general approach to structuring data
- The design structure of the library
- The organizing principles of the individual manual.

Task-Oriented Approach

When writing a computer manual, the most important question a writer asks is: how do I structure my book? According to which principles do I organize the available material? Which items do I include and which should I omit? What information should I provide at what time?

VSE uses a **task-oriented** approach to organize and structure information in a way that both experienced and inexperienced users of a product can easily follow.

The basic question a writer of task-oriented manuals asks is:

- Which tasks do I perform with this product?

The information given then in the manual is grouped to support that task. All the information in a task-oriented book should be appropriate for the task being performed. Appropriateness to task is the standard for excluding or selecting information.

The task-oriented approach makes for the design structure of a good product library.

General Documentation

Product information should reach customers (managers, system administrators and other data professionals) in two stages: at product announcement in form of a general information manual geared primarily to potential customers and at product shipment in form of operational manuals that give detailed product information.

When writing computer manuals, some companies do not provide a general information manual, arguing that the customer finds out about the product once the product is ordered. Those companies overlook that customers may not be willing to order a product whose applicability can not be evaluated beforehand.

General Information Manual (GIM): The General Information Manual is provided at product announcement. It gives an overview of the product's capabilities and features to help prospective customers and users evaluate the applicability of the program to their installation. In general, a GIM contains the following sections:

- A description of a program's functions, the value of its features oriented to the user's needs and applications, as well as a description of what the end user can do with the program.
- A description of the environment in which the program operates, prerequisite programs, as well as required hardware and software resources.
- If applicable, a GIM can cover:
  - Additional optional software

- Performance considerations
- Conversion considerations
- Customer responsibilities
- A description of the tasks that can be performed with the program.
- The titles of supporting publications, stating the intended use of the publication and summarizing the information contained in those manuals.

Shipment Documentation

Operational Manual(s): The operational manual(s) is/are structured according to the different tasks to be performed. The library may include the following manuals:

- The **Planning** manual is written to guide the customer on the options a program offers and the resulting decisions that need to be made for the tasks of installation, customizing, operation, administration, application programming, and diagnosis of the software.

  Topics covered may include:

  - Installation planning to the product. For example, which system configurations support the program?
  - Operation planning to the product. For example, which resources does the program use, or which abnormal events might occur during execution of the program?
  - Customizing planning to the product. For example, which user routines must be appended to the program?
  - Migration planning to the product. For example, which conversion aids can be used to migrate to the program?
  - Administrative planning to the product. For example, which resources (databases, transactions, or user profiles) must be defined by the program?
  - Application programming planning. For example, which system services or resources are required to support application programs?
  - Diagnosis planning. For example, which diagnostic aids are available?
- The **Installation** manual describes how to install and customize the product so that it is ready to be used.
- **Operations** should describe the facilities for the user, the purpose and the means.
- **Messages and Codes** lists all messages and other error codes together with the reason and the appropriate actions to take. This could also be a chapter in the Diagnosis manual.
- **Diagnosis**, either a separate manual or a chapter, should guide the user/administrator to determine if a problem is a user error, or a problem within the product or a problem of a product combination.

Library Design

Manual Structure

A good publication consists of the following parts and structure:

- **Front Matter**. It consists of Preface, Table of Contents, a Bibliography that explains where the reader can find additional information introduced in the manual and, if applicable, Summary of Amendments.
- **Main Part** or Body of the Book. It consists of various chapters that develop the topic of the manual.
- **Appendix**. If applicable, the appendix consists of samples and items that supplement the content of the manual.
- **Back Matter**. It includes a glossary followed by an index.

Packaging Considerations

After you have completed your task analyses and decided what information you must supply for each task, you are ready to consider how to package the information. Here are some questions that guide you with that process:

- What should be the size of the publication? As a general rule, do not publish books of more than 200 pages or less than 20 pages.
- Which word processing system should I use? IBM BookMaster 3.0 provides all the modern publishing support needed.
- Should I deliver hardcopy or softcopy information? Having information on CDROM has been very well received by customers.

More Information

For information relating to the publication process and producing better quality publications, refer to the IBM guides:

- *Producing Quality Technical Information*, SC26-4195.
- *A Practical Guide to Consistent Books*, ZZ27-7325.

Chapter 5. Providing National Language Support (NLS)

In today's worldwide marketplace, products must be designed to provide support for national languages.

When the subject of National Language Support is mentioned, most people think of translating software panels, messages, and publications into languages other than English. But, National Language Support consists of much more than just translation. To really support a national language, a product must accommodate all aspects of the language. It demands attention to sorting sequences, to date and time, to currency, and to many other national and cultural factors, to mention just a few items.

IBM has published a set of rules and guidelines that are considered necessary to enable a product to provide that support. Like all other design considerations, the decision to follow particular rules and guidelines is made after balancing various requirements with design constraints. For your program to achieve maximum benefit from these rules and guidelines, however, you should consider them at the earliest possible development stage.

Statements regarding National Language Support for any country named in these rules and guidelines are suggestions for general applications. Particular industries and applications may require National Language Support different from what is stated here.

Common User Access (CUA)

The Common User Access (CUA), a basic element of Systems Application Architecture (SAA), ensures consistency in designing interactive user interfaces. To ensure that proper National Language Support can be provided to the interfaces developed under this architecture, all IBM publications mentioned in this description can be used in conjunction with the CUA documentation.

Concepts of National Language Support

There are two concepts vital to an understanding of National Language Support: *enable* and *implement*.

Enable

To *enable* a product means to design that product in such a way as to *facilitate* the inclusion of national language functions and to design a product in such a way as not to inhibit the inclusion and usability of national language functions in other products.

During the design stages of a product, the developer must keep in mind that the architecture and the design have to be flexible enough to allow national language functions or any specific language to be included in the product (possibly at a later time) without requiring a redesign.

Implement

To *implement* a national language on a product means to add national language functions to a product when the design has been *enabled* to accept those functions, as well as providing the customer and service information in the user's national language. Implementation refers to specific languages and to specific additions to a product.

Language Subsets

If languages are divided on the basis of their implementation characteristics, the rules and guidelines fall into subsets that naturally support those characteristics. By using all the subsets and guidelines, a designer ensures that a product is enabled for the national languages of all countries [1].

The following characteristics form the basis for creating subsets of the enabling rules and guidelines for written languages:

- Left-to-right languages using single-byte characters. For example, Danish, English, Finnish, French, German, Hungarian, Italian.
- Bidirectional languages using single-byte characters. For example, Arabic, Hebrew, Persian.
- Languages using double-byte characters. For example, Chinese, Japanese, Korean.

These groupings are based on practicality in data processing rather than linguistic reasons. For example, the bidirectional languages are set apart from the others by their common requirement for functions that permit handling of entry and presentation in two directions (right-to-left and left-to-right). Double-byte languages are distinguished by their requirement for more than one single byte per character to code their large character sets.

The first subset applies to left-to-right single-byte languages and contains the rules and guidelines that are common to all languages. It is considered a base for all products. The other subsets deal uniquely with the bidirectional and double-byte languages.

National Language Standards and Laws

There are many language-related standards and laws. Most countries have language laws affecting the importation, sale, or use of data processing equipment, software, or documentation. Some of the laws specify the language(s) to be used on labels, keyboards, documentation, or software. Other laws regulate cultural aspects such as date formats, calendars, or numeric representation. Because there are many laws and constant revision of laws, only the country issuing a law can describe it [2].

Some countries like Canada, Switzerland, and Belgium have legal requirements to support more than one official language.

Implementation Considerations

If you plan to include National Language Support in your software package, you have to consider many design and implementation aspects, for example:

- Which code page(s) are required;
- Which character sets are required;
- Isolation of language dependent hardware, and software code from executable code at the source, load module, and packaging levels;
- Definition of graphic characters as delimiters for control purposes;

---

1 For a list of countries refer to *IBM National Language Information and Design Guide, Appendix B,* form number SE09-8001.

2 For more details refer to *IBM National Language Information and Design Guide, Appendix C,* form number SE09-8001.

- Definition of variables used in translatable material and its location and order within a field;
- Expansion space for translation (expansion is based upon complete words and blanks associated with those words; up to 10 characters English length require 100-200% while 70 characters or more English length require only 30% expansion space).
- Inputs such as commands, keywords and responses must accept English inputs in the same session with the NL inputs.

National Language Support for VSE/ESA Version 2

VSE/ESA Version 2 currently offers English, German, Spanish, and Japanese for following products/functions:

- Interactive Interface
- Online Message Explanation (OME)
- DWF
- Selected CICS end-user messages (not Spanish)
- High Level Assembler for VSE messages

For the interactive interface, the OME, only one language can be used per installation. The customer chooses the language when ordering the product.

The interactive interface of VSE/ESA is comprised of panels, help panels, online messages and help messages.

The OME allows to display the explanation of any message issued by one of the VSE/ESA base products in the customer selected language.

VSE Workdesk is a front-end to the interactive interface for workstation users.

DWF is an application development workplace for workstation users. It delivers all available languages, but it is recommended to use only the language that was chosen for the VSE/ESA host system.

CICS allows to select the language

- globally by using CICS initialization options
- for each end-user
- for each terminal or transaction

High Level Assembler for VSE allows to select the language for each assembly run.

Some end-user applications, IBM DisplayWrite/370 for example, provide multilingual support. **All** languages may be installed in the system and the user selects the desired language.

More information

IBM has published a set of rules and guidelines that describe how to provide national language support. For a bibliography, please refer to “IBM National Language Support Manuals” on page 167.

Part 4. Creating Installation Tapes and Servicing Your Product

This part describes how to create installation tapes and how to service your product. Additional samples corresponding to the topics described here are given in Part 5, “Packaging and Service Samples” on page 111.

Chapter 6. VSE Product Numbering Conventions

This chapter explains the IBM VSE product numbering conventions and helps vendors to avoid numbering conflicts at customer installations.

These numbers reflect the structure of your product and are required by the Maintain System History Program (MSHP) to ensure correct installation and service of your product.

MSHP is a program in VSE used for installing and servicing an IBM product. It is used for automating and controlling various installation and service activities for a VSE system. You can use MSHP to install your own products on VSE and to apply service to them.

Important

The structure influences how a product is installed and serviced. Therefore it is very important that you consider the structure of your product early in the development process. Refer to section “Rules for Product Structuring” on page 74 for a description of dependencies between product structure, numbering scheme, distribution tape creation, and installation.

MSHP Product Identification

For installation and service of a program, MSHP uses alphanumeric combinations that identify the product to MSHP. Here MSHP follows the standard IBM coding format for product identification. The identification format consists of the following parts, not necessarily used in this sequence.

Format

```
┌┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┐
╡  TTTT-PPP-CC-CLC   ╞
└┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┘
```

where:

TTTT stands for a four digit numeric code. It is part of the order number and is referred to as **type**.

At present, IBM uses the range 5600 to 5799.

For example, in Figure 6 on page 72, the type is 5686 and identifies a VSE product.

PPP stands for a three letter alphanumeric code. PPP is part of the order number and is referred to as **model**. In the context of MSHP, the model is called **product code**.

The product code ranges from 001 to 998.

For example, as shown in Figure 6 on page 72, the product code for VSE Central Functions Version 6 is 066.

Note: MSHP accepts also alphanumeric combinations.

CC stands for a two digit numeric code, the **component number**, indicating the component of a product.

The component number ranges from 01 to 98 within one product.

For example, as shown in Figure 6 on page 72, MSHP is the seventh component of VSE Central Functions.

CLC refers to the **Component Level Code**. The CLC identifies the release of a product within a version.

For example, in Figure 6 on page 72, the component level code for components belonging to VSE Central Functions Version 6 Release 1 is 15C.

The following example shows numeric combinations that identify a product according to product name, product number, component ID, CLC, and product identifier to MSHP.

Note: This format is used for all IBM VSE products uniformly since VSE/Advanced Functions 2.1.

Figure 6. Product Identification Convention Example

IBM Product	IBM Product Number	Component ID	CLC	Product ID (MSHP)
VSE/Cen. Func. 6.1.0 • VSE/POWER • MSHP • VSE Workdesk	5686-066	568606603 568606607 568606618	15C 15C 1CM	06615C 06615C 0661CM
CICS/VSE 2.3.0 Production Part	5686-026	568602601	14X	02614X
Display Write/370 2.1.0 Base Product	5686-022	568602201	A10	022A10

MSHP uses some of these numeric combinations for installation, others for service procedures.

To create the distribution tape, both component ID and product ID are needed. The following section describes how these are constituted.

Component Identifier

The component ID identifies the component(s) of a product. For products to be serviced using MSHP, that is to install PTFs and APAR fixes (ZAPs), one component identifier has to be defined per component of a product. The component identifier is built from the program number followed by the component number. The basic format of the component ID thus is TTTT-PPP-CC.

Fully-qualified component identifier

In order to identify the component uniquely to MSHP, the component must, however, be specified in "fully qualified" format. Fully qualified means specifying both the **component ID** and the **release level** of the component. This is what the complete, fully qualified component ID format looks like:

```
┌┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┐
%   TTTT-PPP-CC-CLC   %
└┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┘
```

The fully qualified component ID for MSHP, for example, is 5686-066-07-15C. This identifies MSHP as component 07 on level 15C of product 5686-066, which means VSE Central Functions Version 6.1.

A product usually consists of one component. Larger products, such as VSE Central Functions, contain several components or functional units; it has as its seventh component, for example, MSHP (Maintain System History Program). For products that contain more than one component, refer to “Rules for Product Structuring” on page 74 for MSHP requirements.

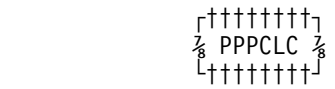
In summary, components refer to the structure of a product, which should carefully be considered. In general, one component should be sufficient, with the possible exception for NLS support.

It is very important that you consider the structure of your products into components in time. The structure influences how a product is installed and serviced. Since PTFs are built for components, a large number of components may cause an increase in corequisite PTFs. For more information on corequisite PTFs, please refer to “Ensuring the Correct Environment” on page 103.

Product Identifier

The product ID identifies a product and its release level to MSHP for installation and service. It is the product ID that MSHP uses when referring to a product.

It is recommended that the product ID is in this format:



For example, the product ID for VSE Central Functions Version 6 Release 1 is 06615C.

The first three characters (066 in the example) are the product code. The remaining characters (15C in the example) identify the release level (CLC) of the program.

Note: MSHP supports multiple levels of a program. Note that:

- The PPP is changed with a new product version.
- The CLC is changed for a new product release.

Figure 7. Changed Product ID through New Version and Release Level

IBM Product	VRM	Product Number	CLC	Resulting Product ID
DisplayWrite/370	1.1.0	5666-338	H82	338H82
DisplayWrite/370	1.2.0	5666-338	D58	338D58
DisplayWrite/370	2.1.0	5686-022	A10	022A10

Note: VRM = Version Release Modification

Using the Component and Product Identifier

When writing MSHP jobs using the identifiers described above, follow these rules:

1. Use the product ID rather than the CLC alone if both specifications are possible in an MSHP statement. This holds true, for example, for the PRE parameter of the REQUIRES statement. Because CLCs are not always unique, a duplicate CLC might prevent application of a PTF or installation of a product; MSHP can not distinguish the different products using the same CLC.
2. Always use the fully-qualified component identifier in order to refer to a component on its correct release level.

Note

The identifiers used for product installation can't be changed by service.

Rules for Product Structuring

The following MSHP requirements must be considered when defining components:

- If a program consists of a group of components that **must be installed together** (like some components of VSE Central Functions), all components must have different component IDs, but the same CLC (and therefore the same Product ID). To create an install tape, the product (residing in one sub-library) is first defined to MSHP with one ARCHIVE statement for each component and one for the Product ID and then put to tape with one BACKUP Product ID statement. The product is installed with one install job.
- If a program consists of a base component (group of components) and a series of features, out of which **only one** can be selected (for example, a NLS feature), all components of the features can have the same component ID but must have different CLCs (one Product ID for the base and one for each feature). To create an install tape, the base and each feature (residing in different sub-libraries) are defined to MSHP with corresponding ARCHIVE component ID and ARCHIVE Product ID statements and are put to tape with separate BACKUP Product ID statements. The base and each feature are installed with different install jobs.
- If a program consists of a base component and a series of features, out of which **one or more** can be selected (for example, additional functions), all components must have different component IDs and different CLCs (one Product ID for the base and one for each feature). To create an install tape, the base and each feature (residing in different sub-libraries) are defined to MSHP with corresponding ARCHIVE component ID and ARCHIVE Product ID statements, and are put on tape with separate BACKUP Product ID statements. The base and each feature are installed with different install jobs.
- If a component of a feature replaces a base component, the same component ID must be used for the base and for the feature, but the CLC must be different. At installation time MSHP issues a warning that the existing part will be overwritten.

Refer to Chapter 7, “Creating Installation Tapes” on page 79 for details on creating installation tapes.

Convention for Vendor Product Identification

The purpose of this section is to

- Encourage the use of MSHP also for non-IBM products (makes life easier for customers and IBM service).
- Keep the vendor products from stepping on each others' and on our toes when they invent their own MSHP numbering scheme (which they did).

It is addressed at vendor products, completely independent of any IBM relation. Therefore these rules do not apply whenever a vendor product is subject of a cooperative marketing agreement or even closer relationship.

The previous sections explain the relationship between product number and all the MSHP numbers in the ideal case. That is, either it's an IBM product that has the same program number worldwide or a non-IBM product where the product owners set up their own program numbers, independent of IBM, and also worldwide. Such a program number serves as the base to derive in conjunction with a CLC the Product ID, component ID, and the fully-qualified component ID. In case of a vendor product with cooperative marketing agreements with different IBM country organizations, different IBM program numbers will be assigned in different countries. Then the - usually later - assigned IBM program number(s) will not resemble the component ID, which would be the same worldwide. Anything else would imply differently built distribution tapes for each of these countries.

Each product that works with VSE should have unique component and product identifiers. This applies to both IBM and vendor products. The following naming convention ensures this uniqueness for products worldwide. Identify your products as follows:

Type

The numbers 5600 - 5799 must **not** be used for type, since these numbers are reserved for IBM.

Model

Use a **V** as first character of your model/product code (PPP).

CLC

The CLC value used by IBM VSE products does not contain the letters **T**, **U**, **V**, **W**. Thus, these letters are available to vendors to identify their products. The first two characters of the CLC value identify the vendor as follows (the third character can be chosen by the vendor, it can be A - Z, 1 - 9):

European vendors

TAx ABACO INFORMATICA S.A., Madrid, Spain
TCx ALPI S.P.A., Milano MI, Italy
UTx Alldata Software Haus, Stuttgart, Germany
TDx Archetype System Ltd., Hatters Lane, Watford, Herts WDI 8YH, UK
TEx ARTHUR ANDERSEN AND CO, London WC2R 3LT, UK
TKx Becker Software GmbH, Wiesbaden, Germany
TFx BTB Betriebswirtschaftliche u. Technische Beratungsgesellschaft mbH, Leinfelden-Echterdingen, Germany
TGx Byte Software House S.P.A., Torino, Italy
TJx CAP Debis SHI, Muenchen, Germany
TYx CAP Debis OrganisationsPartner GmbH, Bad Oldesloe, Germany
TTx CGI Interprogramm GmbH, Langenfeld, Germany
THx CIOB, Eindhoven, Netherlands
UHx Comparex, Mannheim, Germany
TIx COTEC - LISTER PETTER LTD, GLOS GL11 HS, UK
TLx ERITEL, Madrid, Spain
TMx Dipl.-Ing. Rainer Gehrke, Unternehmensberatung GmbH, Overath, Germany
TNx H&M System Software GmbH, Roedermark, Germany
TOx Ibias, PL Gouda, Netherlands
TPx IKO Software Service GmbH, Stuttgart, Germany
TQx Infologica, Stuttgart, Germany
TRx Infosoft Deutschland, Weiherhof-Zdf., Germany
TSx Insurance Software & Systems Ltd, Oslo 1, Norway
TUx IPACRI S.P.A., ROMA RM, Italy
TVx Klumpp Informatik, Stuttgart 31, Germany
TWx Lattwein GmbH, Dueren, Germany
TXx LS3, London W1N 7RA, UK
VIx Macro4, Worth, West Sussex, UK
TZx ORBIT COMPUTER SYSTEMS, Manchester, M3 5LF, UK
T1x Osys AG, Zuerich, Switzerland
T2x QUALITY SOFTWARE PRODUCTS LTD LEATHERHEAD, KT22 7AH, UK
T3x SAP AG, Walldorf, Germany
UJx SAPIENS, Israel
TBx Sema Group Systems AG, Wilhelmshaven, Germany
T4x Sema Group, Bruxelles, Belgium
T5x SIO, Montrouge, France
T6x SISTEMI INFORMATIVI S.P.A., ROMA RM, Italy
T7x SLIGOS, Paris La Defense, France
T8x Software AG, Darmstadt, Germany
T9x Soleri - Cigel, Puteaux Cedex, France

UAx STERIA, 78140 Velizy-Villacoublay, France
UBx TIMEGATE COMPUTER SYSTEMS LTD, WALSALL WS9 0QD, UK
UCx UNILOG RESEAUX et Systems, Paris 9, France
UDx Update GmbH, Kulmbach, Germany
UEx USU Softwarehaus Unternehmensberatung GmbH, Moeglingen, Germany
UFx VOLMAC, 3511 GB Utrecht, Netherlands
UGx Wilken GmbH, Ulm, Germany

American vendors

WOx ALLTEL Inc. (Systematics Inc.), Little Rock, AR
VAx Altai Software , Arlington, TX
WAx American Management Systems, Inc., Arlington, VA
WBx American Software, Atlanta, GA
W8x APSIS Software, Inc., Columbus, OH
VBx BI Moyle Assoc., Minneapolis, MN
VCx BMC Software, Sugar Land, TX
VDx Candle Corporation, Los Angeles, CA
WCx Cincom Systems, Cincinatti, OH
VJx Computer Associates Int'l, Islandia, NY
VKx Computer Associates Int'l, Islandia, NY
VSx Computer Associates Int'l, Islandia, NY
WUx Cyborg Systems Inc., Chicago, IL
WDx Data Design Associates, Sunnyvale, CA
W5x Data Kinetics, Ltd., Ottawa, Ontario, Canada
WKx Dun and Bradstreet, Atlanta, GA
WEx Genesys Software Systems, Methuen, MA
WFx Global Software, Raleigh, NC
WVx H & W Computer Systems Inc., Boise, ID
WXx Healthquest, Atlanta, GA
WYx Information Associates, Inc., Rochester, NY
WGx Information Builders, New York, NY
WHx Information Sciences Inc. (INSCI), Montvale, NJ
WIx Information Systems of America (ISA), Atlanta, GA
WZx Integral Systems, Inc., Walnut Creek, CA
VFx IntelliWare Systems, Inc., Dallas, TX
WWx Kirchmann Corp., Orlando, FL
VGx Landmark Systems, Vienna, VA
WJx Lawson Associates, Minneapolis, MN
VEx Legent Corp., Vienna, VA
VQx Legent Corp., Vienna, VA
VHx Mac Kinney Syst., Springfield, MO
WSx Micro Tempus Inc., Montreal, Ontario (Canada)
W9x Open Connect Systems, Dallas, TX
W7x Open Software Technologies, Longwood, FL
W1x Performance Software, Inc., Richmond, VA
WTx Phoenix Software Co., Los Angeles, CA
WMx SAS Institute, Cary, NC
VLx SDI, San Mateo, CA
WRx SDM International Inc., Fuquay Varina, NC
VMx Smartech Systems, Inc., Dallas, TX
W2x Software Diversified Services, Minneapolis, MN
VOx Software Engineering of America, Atlanta, GA
WNx SPSS Inc., Chicago, IL
VNx Sterling Software, Chatsworth, CA
VRx Sterling Software, Chatsworth, CA
VPx Syncsort, Woodcliff Lake, NJ
W6x SysData International, Inc., Hoboken, N.J. USA
WPx Thorn EMI Computer Software, Chelmsford, MA
W4x UNITECH Systems, Inc., Lisle IL USA
WQx Walker Interactive, San Francisco, CA
W3x Xerox Computer Services, Los Angeles, CA

Deviations

Blueline Software uses: 1000-XXX-WE-YYY. 1000 and WE identify Blueline Software. XXX and YYY are internal code different by product and release.

Computer Associates International uses: 0202-CA-...

Legent Corporation, formerly Goal Systems International, Inc. uses: 7965-XXX-00-YYY and 1989-XXX-00-YYY 7965/1989 and 00 identify Legent Corporation, XXX identify the product and YYY the release.

Open Software Technologies uses: 2822-V..-..-W7.

Sapiens uses: 1818-V..-..-UJ.

Software Pursuits Inc. uses: 1975-XXX-01-YYY. 1975 and 01 identify Software Pursuits Inc. XXX and YYY are internal code different by product and release.

Request and updates for CLC numbers

Please use the address listed on the Reader's Comment Form to mail CLC number requests.

Chapter 7. Creating Installation Tapes

This chapter describes how to prepare a distribution tape for a VSE product. Here the objective is:

- To facilitate installation by VSE customers. This requires an understanding of how IBM tapes are installed and serviced by MSHP.
- To simplify procedures for customers familiar with IBM conventions.

Note

The structure of a product defines the layout of a product distribution tape and the product installation and service process. Therefore it is important that the implications of a product structure are already considered as part of the product design. Refer to “Rules for Product Structuring” on page 74 for a description of dependencies.

Creating a Product Distribution Tape on VSE

A distribution tape must conform to the tape layout as described below if you want to achieve the following:

- The product can be installed using MSHP.
- The product can be installed using the VSE product installation and service dialogs.
- The product can be serviced by PTFs.

System Requirements

A distribution tape for installation on a VSE system must be built on a VSE system.

Preparing a Library

Creation of a product tape requires preparation of the VSE library. Follow these steps to prepare the VSE library for installation of your distribution tape:

1. Create your library.

   You can create a library in either of the following ways:

   - As a sequential file. For an example, please see “Creation of a Library on a Sequential Disk Extent” on page 113.
   - **Or** as a file in VSAM-managed space. This lets you use dialogs. For an example, please see “Definition of a Library in VSAM Managed Space” on page 114.

   Note: Creating a library is optional if you use an existing library.

2. Create a sublibrary that contains the code. You can create a sublibrary in either of the following ways:

   - By using the dialogs
   - **Or** by going into the ICCF command mode and using LIBR.

   The name of the sublibrary should be unique for your and also for the customer's installation. Using the following naming conventions ensures uniqueness:

for *production part*: `aaa.PRprod-id`
for *generation part*: `aaa.Gnprod-id`

where:
`aaa=` library name
`prod-id=` product ID
`n =` 1 through 9, depending on the number of generation sublibraries.

3. Compile or assemble the source code of the product. Catalog the object modules into the sublibrary you just created. This sublibrary is the production library. It contains everything that is necessary for running and servicing your product.

   Points to Remember:

   - All members should have names that identify them uniquely as part of the product. This is achieved within IBM by assigning three characters to a product. These characters are called the Component Code. Names of modules, phases, or also a reserved file name for a disk file should start with these characters.

     For example, all VSE/POWER modules start with IPW.

   - If the product includes Librarian source members, select the correct member type from the list in Chapter 11, “Library Member Types” on page 131.

   - If you are not using the High Level Assembler for VSE, executable macros have to be processed with the EDECK option of the assembler. With High Level Assembler for VSE, E-decks are no longer used. However, for compatibility reasons, the EDECKXIT parameter is available, which allows High Level Assembler for VSE to translate E-decks back to A-books before processing.

     Notes:

     a. Macros can only be modified in A-book format.

     b. High Level Assembler for VSE cannot convert A-books to E-decks.

     Refer also to the library member type E in Chapter 11, “Library Member Types” on page 131.

   - Your code should include such items as installation material and data files, because all non-library data has to be handled separately.

Optional:

4.

   - If your program includes a generation part to be compiled on the customer site for better adaptation to the customer environment, carefully catalog this material into a separate generation sublibrary. You may use more than one sublibrary (only if the generation part does not fit on one volume of the smallest DASD supported by VSE); this should be avoided when possible. This generation library need not be installed in the production environment, saving disk space.

     For example, the generation feature of VSE/Advanced Functions includes all macros needed to generate the supervisor.

   - Do not use a generation library for shipping code that is needed for service of the production part.

5. Link your product invoking the linkage editor.

   Compiling or assembling transforms your program into object module(s) that are then cataloged into the sublibary.

   If your program is independent of other programs, you may also link it and ship only phases. Linking at the customer's site is then not required. In this case, only the phase must reside in the sublibrary

used for creating the tape. Note that service must then be done for phases, that is, PTFs must contain phases not object modules.

In case your program needs other programs, for instance CICS modules, you could still link your program with one level of CICS. Then you would ship all phases including those object modules that need relinking with the version of CICS installed at the customer's site. In this case, supply LINK books.

Creating the Header

For each product (installable unit) create a header and catalog it into the sublibrary that contains the code. Follow these steps to create the header and catalog it:

1. Use the letters "HD" followed by the product ID to compose the **header name**, that is, use HDPPPCLC.
2. Write the content of the header file to be as follows:
   - 80 byte records that can contain any text. For IBM products these records contain the copyright records, or a repetition of those records if more than one copyright notice is required.

     The format of the text record is:

     ```
     col 1 - 6      HDR001
     col 7 - 8      sequence number (01 to 99)
     col 9          blank
     col 10-71      text (see example)
     col 73-80      optional card sequence number
     ```

   - One end indicator record.

     The format of the end record is:

     ```
     col 1 - 6      HDR099
     col 7 - 8      sequence number (01)
     col 9          blank
     col 10-12      END
     col 13-71      not used
     col 73-80      optional card sequence number
     ```

   Example of the content of a header file:

   ```
   CATALOG HD066.Z  REPLACE=YES        /* CATALOG HEADER FILE*/
   HDR00101  LICENSED MATERIAL - PROPERTY OF IBM
   HDR00102  5686-066 (C) COPYRIGHT IBM CORPORATION  1995
   HDR00103  ALL RIGHTS RESERVED.
   HDR00104  US GOVERNMENT USERS RESTRICTED RIGHTS -
   HDR00105  USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
   HDR00106  GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
   HDR09901  END
   /+
   ```

   Note: IBM developers should check Corporate Standard C-S 0-6045-002 for the latest wording of the copyright statement.
3. Catalog the header file as a member of type "Z" into your sublibrary using the following job:

```
// JOB HEADER
// EXEC LIBR
   ACCESS  SUBLIB=lib.sublib
   CATALOG member.Z
      ... source of member.Z
/+
/*
/&
```

Creating the History

Once the header has been created, the next step is to create the product's history file using MSHP.

Follow these steps to create a history file:

1. Check that you meet the requirements to create a history file, which are:
   - disk space (non-VSAM)
   - product identification

   For product identification requirements, see “MSHP Product Identification” on page 71.
2. Write the MSHP job to create the history file.
3. Execute the MSHP job.

Example for creation of an MSHP history file:

```
// JOB MUEDHIS
* DEFINE HISTORY FOR DW/370 2.1.0
// ASSGN SYS021,DISK,VOL=DW202T,SHR
// EXEC MSHP,SIZE=1024K
  CREATE HISTORY SYSTEM
    DEFINE HISTORY SYSTEM UNIT=SYS021     -
                       EXTENT=0705:08     -
           ID='DW/370 5686-022 2.1.0 BASIC.HISTORY'
  ARCHIVE  5686-022-01-A10        /* FULLY-QUALIFIED COMPONENT-ID */
  ARCHIVE  022A10                 /* PRODUCT-ID   */
    RESOLVES 'DW/370BASE.2.1.0 -  5686-022'
    COMPRISE 5686-022-01                                  -
         PHASES=(DDD*,DKL*,EDD*,EKL*,T1D*,$$$CO*)         -
         MODULES=(DDD*,DKL*,EDD*,EKL*,$$$CO*)             -
         MACROS=(DDD*,EDD*,$$$CO*)                        -
           TYPE=A
    COMPRISE 5686-022-01                                  -
         MACROS=DDD*                                      -
           TYPE=E
    COMPRISE 5686-022-01                                  -
         MACROS=(DDD*,HD*)                                -
           TYPE=Z
    COMPRISE 5686-022-01                                  -
         MACROS=(CLI*,DW*,EMPT*,HEX,DKL*,EX*)             -
           TYPE=X
  RESIDENCE PRODUCT=022A10           -
         PRODUCTION=DW202DA.PR$A10
/*
/&
```

Figure 8. Creating a History File

This MSHP job contains the following major parts:

CREATE HISTORY

The extent information for the history file is supplied in either of the following ways:

- Using the MSHP DETAIL CONTROL statement with DEFINE (as shown in the sample)
- **Or** using the DLBL and EXTENT statements for file name IJSYSHF

ARCHIVE component

This ARCHIVE statement describes your component to MSHP.

In our example the component is: 5686-022-01-A10 .

ARCHIVE product

This ARCHIVE statement describes your product and is used to enter this information into the history file.

In our sample the product ID is: 022A10.

RESOLVES The MSHP RESOLVES statement associates here a comment with a product.

In our example the comment is: DW/370BASE.2.1.0 - 5686-022.

The first 16 characters of the RESOLVES statement must correspond to the

tapefile ID as specified in the BACKUP job and is coded as shown on page on page 85.

The first 16 characters of this comment are used by the Service Dialogs as a nickname for the installed product.

COMPRISE The COMPRISE statement is used to specify the component, phases, modules, and/or macros that make up a product. That information is entered into the history file.

The above sample uses generic COMPRISES rather than listing each member separately. Use generic COMPRISES whenever possible. The definition must, however, be unique across all products.

COMPRISE statements for macros are repeated for different macro types. Please refer to Chapter 11, “Library Member Types” on page 131 for a list of macro types that are allocated for specific use.

Note: Procedures and library members of type .PROC cannot be specified and serviced by MSHP. But you may ship them in the sublibrary of your product.

RESIDENCE

The RESIDENCE statement defines the names of the production sublibrary or production and generation sublibraries in which a product resides. This information is recorded in the history file for any follow-on activities, such as service application, installation, or product backup.

Restrictions:

- Use the INVOLVES LINK command only if the user does not have to relink after, for example, compiling some modules or adding other libraries. Else, if a link is required for product activation, a LINK job should be provided and mentioned in your Program Directory and/or Installation manual.
- Reduce requisite information to a minimum. Do not build MSHP REQUIRES statements into your job except when creating the history file for a feature.

For additional samples of how to build a history file, see “Creating or Changing VSE/Advanced Functions History Information” on page 117.

Creating the Tape

Now that you have prepared everything for building the tape, this is how you create the tape:

1. Write your MSHP BACKUP job. For a sample see below.
2. Execute the job.

Example for an MSHP BACKUP job:

```
// JOB MUETBASE
// ASSGN SYS021,DISK,VOL=DW202T,SHR            /* DW/370 HISTORIES */
// PAUSE TAPE M1669 ON 570 MOUNTED ?  ---> PRESS ENTER, IF MOUNTED.
// ASSGN SYS006,570,D0
// EXEC MSHP,SIZE=1024K
   BACKUP  PRODUCT=022A10                    -
           ID='DW/370BASE.2.1.0'             -
           HEADER=HD022A10                   -
           PRODUCTION
     DEF HISTORY SYSTEM                      -
             EXTENT=0705:08  UNIT=SYS021     -
             ID='DW/370 5686-022 2.1.0 BASIC.HISTORY'
/*
/&
```

Figure 9. MSHP BACKUP Job

The MSHP BACKUP job contains the following major parts: JCL and BACKUP PRODUCT.

JCL

- Assign the distribution tape as SYS006.
- Provide ASSGN, DLBL, and EXTENT for the system history file. Usually, this is already available in the partition setup.
- Provide ASSGN, DLBL, and EXTENT for the selected library and LIBDEFs.

BACKUP PRODUCT

This statement is used to create the tape. It writes the header, the history file, and the product sublibrary onto tape.

The BACKUP PRODUCT statement has the following values:

- The value of PRODUCT is the product ID as used in the ARCHIVE product statement when creating the history file in Figure 8 on page 83. In the sample this value is 022A10.
- The value of ID is the tapefile ID and is coded as follows:

  It is 16 characters long. The first characters contain the product name or a suitable abbreviation of it; the product name is followed by a "." (dot), followed by the product's version, release, and modification level. Spaces not occupied by a character **must** be filled up with "." (dots).

  In general, the first 10 characters contain the product name; the eleventh character is a "."(dot); the last 5 characters indicate the product's version, release, and modification level, separated by "."(dots).

  In the sample this value is DW/370BASE.2.1.0

  If the product's version, release, and modification level occupy more than 5 characters, the product name must be abbreviated. For example, XY/370BA.27.53.0 .

  WARNING: The tapefile ID must **not** contain any blanks.

  Note: The tapefile ID is used by the installation dialogs. It may be specified in an INSTALL job and should correspond to the first 16 characters of the RESOLVES statement when creating the history file.

- The header is the one created for this product. In the sample the header is HD022A10.
- The statement specifies the type of the library to be copied. In the sample it is PRODUCTION only.

Optional: Specify the DEFINE HISTORY statement only if you do **not** have the DLBL and EXTENT statements for the history file (IJSYSHF) in your standard label sets. The file information is the same as specified when creating the history file, see Figure 8 on page 83.

Resulting Tape Layout of the Product

As a result of the BACKUP job, the product is written to tape in the following format:

Figure 10. Layout of a Distribution Tape

File Number	Content	Sample
1	header file	HD022A10
2	product history file	history file for product 022A10
3	product libraries	DW202DA.PR$A10
4	null file (tape mark)	null file
5	EOB (end of BACKUP information)	EOB
6	null file (tape mark)	null file

File 3 of the product distribution tape contains library data only.

Related installation material and other material should be packaged as library data, if possible. The VSE Librarian provides support for shipping executable, parameterized procedures within a VSE library, eliminating thus the need to package customizing and activation jobs as non-library data.

Shipment of Non-Library Material

Products may ship non-library material on a non-stacked distribution tape as additional files **after** the library/history BACKUP copy. Examples are:

- Copies of VSAM files
- Machine readable documentation
- Non-serviced material.

Observe the following for shipment of the distribution tape:

- Installation of these additional files on the distribution tape is not supported by the install dialogs.
- Installation of such additional files on the distribution tape must be possible through functions supplied by either of the following:
  - VSE/Advanced Functions (for example, system utilities)
  - The product itself (for example, ICCF DTSUTIL function for ICCF library)
  - Products that are prerequisite to the product
- At present, no service method exists for these extra files. In addition, MSHP does not control the content and level of such files.

  To solve this, the material of some products is shipped as source statement library members. For example, ICCF library members are shipped as I-books and then moved into the appropriate dataset at installation time.

Creating a Feature Tape

A feature is a separate unit requiring a base product for installation. For identification to MSHP, a feature uses the following numbering convention:

- A feature uses the same program number as that of the base product.
- A feature is identified by its own, unique component level code (CLC).
- A feature has its own component ID. Otherwise, MSHP would treat the feature as a new release of the base product.

The following example illustrates the numbering convention for a base product in relation to one of its features.

Figure 11. Coding Convention Example for a Base Product and a Feature

	Component ID	**CLC**	**Product Code**	**Product ID**
Base Product	5686-007-01	C44	007	007C44
Feature	5686-007-02	C45	007	007C45

Warning: This definition of a feature is strictly in terms of MSHP. Feature and feature code as used for distribution or ordering may consist of one or more features by MSHP definition.

The distribution tape for a feature is created in the same way as for a product, described in section “Creating a Product Distribution Tape on VSE” on page 79. The only difference is when creating the history file. The statement **REQUIRES PRE=prod-id of the base product** is added to the job. For an example, see “History Information for a Feature of a Product” on page 119.

Creating a Tape for Selective Installation of a Product or Feature

At times, a customer wants to select parts of a product rather than installing the entire product on the distribution tape. For example, translated Machine Readable Information (MRI) consists of panels, helps, and messages translated into 15 or more languages. Installing all of the languages occupies a large amount of disk space. Thus, in an attempt to save disk space, a customer may choose to install only some of the languages offered.

While the customer may want to install some selected languages, the product owner may prefer to ship all languages on one tape. The following approach describes how you can build a distribution tape satisfying both needs. Requirements are:

- Each part of the product that can be selected must be a feature according to MSHP standards:
  - The component identifier(s) of these features must be different from the component identifier of the base product. Likewise, every feature must have a unique CLC.
  - Whether one component identifier is used for all of the features or whether every feature has its own identifier depends on the usage:

    If the user should be able to select more than one of the different selectable features, then each of them must have unique component identifiers.

    Note: MSHP checks the component identifier when executing the INSTALL command. If it finds the same component identifier with a different CLC installed, it assumes a new release of the same product and overwrites the previous information. Because of this, the customer

can select only one feature from the offered choice of, for example, language support. This is contrary to the multilingual option offered by some products.

The recommended approach for such cases is, therefore, to have one component identifier per selectable feature. This enables a customer to select more than one, but not necessarily all, of the available language support.

The tape layout is the same, whether the different features comprise the same component identifier and different CLCs, or whether the component identifiers are also different.

- Each feature must reside in a separate sublibrary, have its own header and history file.

Create the history files in either of the following ways

1. Write one job per feature following the steps described in “Creating the History” on page 82 and execute the job.

   Or

2. Join all these separate jobs as job steps into one job. Use the /& only at the end of the complete job. Execute the job.

For a sample see “History Information for a Product Consisting of Multiple Components Installed Selectively” on page 123.

Create the tape

1. Write the MSHP BACKUP statement for backup of the first feature and end it by a /*.
2. This should be followed by the job step for the backup of the next feature.
3. This should be repeated as often as necessary.
4. Put a /& at the end.

For a sample job, see “Back up a Product or Feature for Selective Installation” on page 127. Note that you follow the same steps as for creation of a single product, except that the End-of-Job statement occurs only at the end.

As a result of the BACKUP job, the product is written to tape in the following format:

Figure 12. Tape File Content for Selective Installation

File Number	Content
1	header language 1
2	history language 1
3	library-backup for language 1
...	...
3m + 1	header language m
3m + 2	history language m
3m + 3	library backup for language m
3m + 4	null file (tape mark)
3m + 5	EOB (end of BACKUP information)
3m + 6	null file (tape mark)

Shipping VM Code with a VSE Product

Products may consist of two parts: the program to run on a VSE host and a counterpart in VM. VM files: VSE/ESA Unique Code (UC) uses .Z for this purpose. UC prepares the members by doing the following:

- Diskdump the program to a user's reader.
- Load the members with READ so that they are on disk in disk dump format.
- Wrap the catalog statement around them and catalog them as Z members in the VSE library.
- Transfer the members from VSE to VM as described in the *Installation* manual, SC33-6604, under "VM/VSE Interface" in the index; VSE supplies a skeleton in ICCF 59 SKVMVSE that uses DITTO.

If you used the same approach, you need to ship your skeleton with member type .I and appropriate ICCF commands wrapped around it. ICCF lib 62 is used for optional products.

Shipping PC Code with a VSE Product

Products may consist of two parts: the program to run on a VSE host and a counterpart program to be executed on a Personal Computer (PC). For these products VSE supports distribution of the PC code as part of the product on a normal VSE distribution tape. Then it is possible to download this program on to the PC when the product has been installed on VSE. Depending on the VSE system your product is intended for, you have several possibilities to download your product onto the PC. For more information see Chapter 14, “Shipping PC Code with VSE” on page 145.

Tape Stacking

In order to reduce the number of tapes to be shipped, IBM stacks more than one VSE Optional Product on one tape. Whenever you want to package different products into one offering, you can use this method of stacking tapes.

Stacked tapes can be installed using the installation dialogs of VSE.

Format of a Stacked Tape or Cartridge

The format of a stacked tape or cartridge is as follows:

- A special header (start-of-stacked-tape indicator) comes before the first product on the tape.
- Each product tape appears as previously described; one product following the other.
- A special trailer (end-of-stacked-tape indicator) appears after the last product on tape.

The following figure shows the layout of a stacked tape or cartridge:

Figure 13. Stacked Tape and Cartridge Format	
Header	tape mark start-of stacked-tape indicator record (see description below) tape mark
Product 1	FILE 1 - HEADER FILE FOR FIRST PRODUCT tape mark FILE 2 - MSHP HISTORY FOR FIRST PRODUCT tape mark FILE 3 - PRODUCT LIBRARY FOR FIRST PRODUCT tape mark FILE 4 - NULL FILE tape mark FILE 5 - END OF BACKUP RECORD FOR FIRST PRODUCT tape mark FILE 6 - NULL FILE
Following Products	... ... Repeat files 1-6 for all products to be stacked ... ...
Trailer	tape mark end-of-stacked-tape indicator (see description below) tape mark
End of File	tape mark

Format: The format of the START OF TAPE and END OF TAPE indicator is as follows:

1. tape mark
2. 80 byte record
3. tape mark

Figure 14. Layout of the 80 Byte Record for START OF STACKED TAPE Indicator		
Byte	**Type**	**Content**
00-01	hex	x'0050'
02-03	hex	x'0000'
04-07	hex	x'00000001'
08-44	char	c'START.OF.STACKED.TAPE.FOR.VSE/SP.ONLY'
45	char	c' ' (1 blank)
46-79	hex	34x'00'

Figure 15. Layout of the 80 Byte Record for END OF STACKED TAPE Indicator		
Byte	**Type**	**Content**
00-01	hex	x'0050'
02-03	hex	x'0000'
04-07	hex	x'00000001'
08-42	char	c'END.OF.STACKED.TAPE.FOR.VSE/SP.ONLY'
43-45	char	c' ' (3 blanks)
46-79	hex	34x'00'

The format of the two indicators is the same. The difference consists in the text of the character string from byte 8 to byte 45.

Product Stacking Requirements

Observe the following when creating a stacked tape:

- Use multiple stacked tapes or cartridges only if the number of products do not fit on one physical tape or cartridge.
- If the number of products requires more than one stacked tape or cartridge, products must not span tape or cartridge volumes. Each stacked tape or cartridge must contain the "Start of Tape Indicator" and the "End of Tape Indicator".
- A product requiring another product (which means REQUIRES PRE=prod-id) must follow the required product on the tape. Thus, a product must be stacked before any of its feature(s).

Creating a Stacked Tape

VSE Optional Products are stacked by the IBM distribution centers. For non-IBM products the tape/cartridge stacking process is achieved in either of these ways:

Method 1: Produce a stacked tape by creating the tape.

1. Copy the START OF TAPE indicator to your stacked tape or cartridge following these steps:
   a. Write a tape mark
   b. Write the begin indicator record
   c. Write a tape mark

   Note: Do not rewind the tape.
2. Copy the program(s) to your stacked tape or cartridge. Here follow these steps:
   a. Write the first product to tape using the same back-up job you would also use for creating this product's distribution tape.

      Follow this using the job control statement **// MTC FSF,SYS006,2**. The BACKUP function of the Librarian positions the tape at the end-of-block file written last.

      Do not rewind the tape.
   b. Repeat this procedure for the next product(s) as often as necessary.

c. Copy the END OF TAPE indicator to your stacked tape or cartridge. Here follow these steps:

   1) Write a tape mark
   2) Write the end indicator record
   3) Write a tape mark

d. Write a tape mark.

Method 2: Produce a stacked tape by copying the different parts.

1. Prepare the different distribution tapes as described under “Creating a Product Distribution Tape on VSE” on page 79.
2. Get the correct header and trailer from an existing stacked tape.
3. Use DITTO to copy header, products, and trailer in the correct sequence.

Summary

Consider this for installation and service of your product:

- Have your tapes conform to the IBM numbering convention standards for product identification.
- Have your tapes conform to the IBM distribution tape layout.

More Information

Refer to the following IBM manual for detailed information on JCL and MSHP statements:

- *VSE/ESA System Control Statements*

Chapter 8. Installing and Customizing Your Product

Installation

This chapter describes the VSE installation tools for tapes built according to the specifications outlined in Chapter 7, “Creating Installation Tapes” on page 79.

There are two ways to install your tape on VSE:

1. You can install your tape using the VSE installation dialogs.

   or

2. You can install your tape writing your own MSHP INSTALL job.

Both ways require you to decide which library and sublibrary/ies should receive the product. Both approaches are outlined below.

Using the VSE Installation Dialogs

The VSE dialogs support the installation process, regardless of whether the distribution tape contains one product, one product from which parts can be selected, or several products on a stacked tape-- as long as the tape is built as described in Chapter 7, “Creating Installation Tapes” on page 79.

For detailed installation information refer to the Installation manual that corresponds to the level of VSE that is installed.

Use the following panel selections to install your tape on VSE:

For a tape with one product (layout is shown in Figure 10 on page 86):

- VSE/ESA Version 2

  Install Programs - V2 Format*
- VSE/ESA Version 1 and earlier

  Install Programs - Non-Stacked V2 Format or V1 Format

For a tape with one product, but parts to be selected (layout is shown in Figure 12 on page 88):

- VSE/ESA Version 2

  Install Programs - V2 Format*
- VSE/ESA Version 1 and earlier

  Install Programs - Stacked V2 Format*

For a stacked tape (layout is shown in Figure 13 on page 90):

- VSE/ESA Version 2

  Install Programs - V2 Format
- VSE/ESA Version 1 and earlier

  Install Programs - Stacked V2 Format

Note: ***** indicates that you should ignore the return code caused by message "IESI0083I", which tells you that a nonstacked tape was used.

The dialogs offer a library and sublibrary for product installation. You can overwrite the named library with your own selection of another existing library and/or sublibrary.

Using an MSHP Install Job

If you install a product from tape using an MSHP install job, follow these steps:

1. Select or create a library and select or create a sublibrary in which the product is to be installed.

   For a description see “Preparing a Library” on page 79.
2. Copy the MSHP install job as provided in the product's Program Directory or in the Installation manual and tailor it to the requirements of your installation.
3. Execute the job.

Installing a Tape with One Product: The install job consists of job control and MSHP statements.

- For job control:
  - Assign the distribution tape as SYS006.
  - Provide ASSGN, DLBL, and EXTENT for the system history file. Usually, this is already available in the partition setup.
  - Provide ASSGN, DLBL, and EXTENT for the selected library and LIBDEFs. Usually, this is already available in the partition setup.
- For MSHP statements, use the **INSTALL PRODUCT FROMTAPE** statement with the following parameters:

  PRODUCTION INTO=lib.sublib
  specifies the name of the library and sublibrary.

  The value of PRODUCTION INTO is the name of the sublibrary the product is written into.

  In our example (Figure 16 on page 95), the value of PRODUCTION INTO is DW210A1.CLRLIB2 .

  GENERATION INTO=
  specifies the name of the library and sublibrary package.

  In our sample no generation library is needed.

Note: If you do not specify a library, MSHP takes the library or sublibrary used in the RESIDENCE statement as default.

Optional:

ID= tapefile ID

If used, the value of ID must be the same as coded in the BACKUP job creating this tape. MSHP scans the tape (forward only) for this ID.

If not specified, MSHP restores the product found on the tape on SYS006.

In our example, the value of ID is DW/370BASE.2.1.0 .

Example:

```
// JOB MUEINSTB
* INSTALL THE DW/370 2.1.0 BASE FROM TAPE, NO ADD. DICTIONARIES
// PAUSE TAPE WITH DW/370 2.1.0 BASE MOUNTED
// ASSGN SYS006,570
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K
INST PRODUCT FROMTAPE ID='DW/370BASE.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
/*
/&
```

Figure 16. Installing from a Tape with One Product

Installing a Tape with One Product but Parts to be Selected: For this selective install, you can use the same job as used for a tape with one product. In this case, however, you **must** write one install statement per part selected including the tapefile ID.

The INSTALL statements must be placed so that all products are installed in the same sequence as stored on the tape, if they are to be installed in one run. MSHP scans the tape for the specified tapefile ID, but only in forward direction.

Products that are REQuired by other products **must** be installed prior to the REQuiring product. This explains the requirement to BACKUP a product in the correct installation sequence.

Example:

```
// JOB MUEFRA2
* INSTALL THE DW/370 2.1.0 BASE FROM TAPE, 2 ADDITIONAL DICTIONARIES
// PAUSE TAPE WITH DW/370 2.1.0 BASE MOUNTED
// ASSGN SYS006,570
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K
INST PRODUCT FROMTAPE ID='DW/370BASE.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-LEGA.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-FRA2.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
/*
/&
```

Figure 17. Installing a Tape with One Product with Three Parts Selected

Installing from a Stacked Tape: When installing products from a stacked tape, use the same job as used for a tape with parts that are selected including the tapefile IDs. In this case, however, MSHP is called using the following statement:

```
//  EXEC  MSHP,SIZE=1024K,PARM='PIDSTACKED'
```

Example:

```
// JOB MUESTACK
* INSTALL THE DW/370 2.1.0 BASE AND TWO DICTIONARIES, INSTALL
* CANADIAN FRENCH SUPPORT, FROM A VSE/ESA OPTIONAL PRODUCT TAPE
// PAUSE STACKED TAPE WITH DW/370 2.1.0 BASE AND NLS MOUNTED
// ASSGN SYS006,570
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K,PARM='PIDSTACKED'
INST PRODUCT FROMTAPE ID='DW/370BASE.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-LEGA.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-FRA2.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW/370FRA2.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
/*
/&
```

Figure 18. Installing Products from a Stacked Tape

Customizing

Most products require customizing after having been installed from the distribution tape. Customizing means tailoring the product after installation to the customer's specific requirements. The following is a likely list of items to be considered for customizing.

Avoiding Customer Compilation of Source Code

A software product should be delivered in compiled format regardless of whether or not the source code is delivered. If the source code alone is delivered, the compilation of the source code requires unnecessary time and effort from the customer. Moreover, if you ship source code only, the service must be done in form of the source code; again, this requires customers to compile and link.

Certain customers want to change the source code to meet their needs, in which case a recompilation of certain modules by the customer might be necessary. Keeping this possibility in mind, it is useful to provide customers with the right parameters and exit routines.

Customizing Tasks for the Product

Once the product is installed, the customizing tasks should be performed. The following is a typical list of such tasks.

- Defining and loading one or more VSAM files
- Creating or updating CICS tables
- Setting up the InterSystems Communications (ISC)
- Setting up the user profiles. This includes use of translated panels and messages of one or more languages
- (Re)linking the product
- Adapting the startup job for the product

- Migrating information from an earlier version or release of the product

Writing Customizing Jobs

Software products should contain jobs that customize the product for different installations.

Beginning with VSE/SP Version 2, the following facilities are available for writing customizing tasks:

- Symbolic parameters can be used as variables in the job control language statements. Values can be assigned to the symbolic parameters by the SETPARM statements. Note that SETPARM statements cannot be used to supply values to non-JCL statements.
- It is possible to follow different sequences of job steps depending on the return codes using the Conditional Job Control Language of VSE/Advanced Functions.
- Chapter 15, "Job for Customizing" on page 157, contains as a sample the first part of a customizing job. This job was actually provided for an IBM product and uses the above mentioned facilities. This sample demonstrates the following points:
  - The sequence of job steps is structured **according to the tasks**.

    Conditional Job Control statements serve to combine the different job steps.
  - The information for the user is put at the beginning as a JCL comment, guiding the user how to adapt the job to this particular installation.
  - Variables are used where the user has to supply the values for their particular installation, for example library names. A table in the comment summarizes name, type, default, and description of the parameters. The necessary SETPARM statements are placed right after the explanations. See the section of the comment, headed Start of Parameter List (User Selection), as shown in Figure 55 on page 159.
  - Values in non-JCL statements that may have to be changed for the customer installation are also summarized, along with information on what and how to change.
  - Defaults are provided for
    1. The installation environment, assuming a VSE system as shipped by IBM, as seen, for example, from the library and sublibrary names.
    2. Reasonable sized disk work files accommodating more than one user.

  The result is an operational product that does not need elaborate calculations and tuning as a first step. Tuning can be performed at a later time if necessary. Different steps of the original customizing job may then be reused.

Use of REXX/VSE

Beginning with VSE/ESA 2.1, REXX/VSE is part of the VSE/ESA Base System and can be used to tailor the VSE/ESA operating system instead of using the VSE/ESA conditional job control language. You can use REXX/VSE in the VSE/ESA batch environment for:

- VSE/ESA operation automation
- Substitution and parameterization for job execution
- Direct communication to the VSE/ESA system console
- Input/output (I/O) operations to VSE/ESA libraries and sequential data sets
- Dynamic creation and execution of VSE/ESA job streams

- VSE/POWER job submission and controlling
- VSE/POWER queue element manipulation
- VSE/POWER command execution.
- JCL command execution
- VSE batch program invocation (LIBR, IDCAMS, LNKEDT, ASSEMBLER,..)
- Invocation of high-level programs such as PL1 or C.
- SYSIPT data routing to REXX stems
- Output routing to a REXX stem
- REXX message routing
- Retrieval of some kind of VSE system information

In addition to the versatile programming language offered by REXX, the following functions of REXX/VSE help you write more flexible and powerful customizing programs than with conditional JCL.

REXX/VSE Functions Available with VSE/ESA 1.3.1:

- Accessing VSE/ESA resources

  REXX/VSE provides the EXECIO command, which is compatible with the EXECIO command of REXX/MVS. You may use EXECIO to control input/output (I/O) operations to and from:

  – VSE library members
  – Sequential files
  – SYSIPT (input only)
  – SYSLST (output only).

  EXECIO can read and write data on the program stack or in REXX "compound variables" (called stems) directly. You can use EXECIO for I/O tasks such as copying information to and from a data set, to add, delete, or update information. A REXX program can read information from a data set to the data stack for serialized processing or to a list of REXX variables for random processing. A program can write information from the data stack or from a list of REXX variables.

- Accessing VSE/POWER queues and executing VSE/POWER commands

  You may switch to the VSE/POWER host command environment using ADDRESS POWER to retrieve and store VSE/POWER queue entries. You may also send VSE/POWER commands and retrieve the corresponding output using the OUTTRAP function.

- Creating and executing VSE/ESA job streams dynamically

  By using the character string manipulation functions of REXX, you can dynamically create VSE/ESA job streams and store them into a VSE sublibrary, a sequential file, or into the program stack. In the latter case, when the REXX program exits and returns to job control, job control executes all JCL statements that have been left in the program stack.

- Submitting and controlling VSE/POWER jobs

  By using the VSE/POWER host command PUTQE, you may submit a VSE/POWER job to another partition and optionally wait a specified number of seconds until the job has been started and completed. The OUTTRAP function provides you with a job completion message containing maximum last return code and other useful information about the submitted job. The VSE/POWER job to be submitted may reside in a VSE sublibrary or in a list of REXX variables.

- Extending the programming capabilities of REXX/VSE

You can write your own external functions and subroutines to extend the programming capabilities of REXX/VSE. You can write such functions or subroutines in REXX or in any programming language that supports the system-dependent interface of REXX/VSE. You can also group frequently used external functions and subroutines into a function package. This allows quicker access but they must all be link-edited into a phase. Interfaces to external functions, subroutines, and functions packages are compatible to REXX/MVS.

REXX/VSE Enhancements Available with VSE/ESA 2.1: The following enhancements have been added to REXX/VSE shipped with VSE/ESA 2.1:

- REXX batch programs can address and issue JCL commands.

  A JCL command issued by a REXX batch program is immediately executed and a return code is provided. Input/output data can be exchanged using REXX stems. Thus, REXX can loop on JCL commands and make very efficient use of the JCL conditional job control support.

- REXX programs can communicate with user consoles.

  For messages and replies the REXX instructions SAY and PULL are available allowing interactive communication in line mode.

- New REXX host command environments for calling batch programs.
  - The called batch program can **read** input data from REXX. Within REXX the data is supplied using REXX stems. This is transparent to the batch program and can be used to replace previous data input using JCL.
  - A batch program can also **write** output data to REXX stems using a service routine. This increases the possibilities for subsequent processing of output data created by user programs.
  - Parameters can be passed in the same way as previously done using JCL. In addition, REXX provides a standard parameter list for COBOL, PL/I, and C/370. With it, multiple parameters can be passed and modified by the called program.

- VSE/ESA Librarian and VSE/VSAM IDCAMS commands can be issued from REXX programs.

  Such commands are executed immediately and return codes are provided. Input/output data can be exchanged using REXX stems.

- Extended access to the VSE/POWER list queue.

  Print output can be read from or written to REXX variables, including print control characters, if required.

- Enhanced message routing support.

  The REXX "message external function" allows to specify an output destination for routing REXX messages. It is also possible to specify that REXX messages are to be suppressed.

- A SYSVAR function has been added to retrieve system information such as partition ID, job name, and job number from the job from which REXX was called.

REXX/VSE Enhancements Available with VSE/ESA 2.1.1: REXX/VSE in VSE/ESA 2.1.1 includes a REXX console function that enables you to automate the operation of a VSE/ESA console.

An easy-to-use VSE/ESA console command environment and new external functions make it easy to write REXX console applications.

The console command environment allows to activate and deactivate one or more VSE/ESA console sessions. Console commands can be imbedded into the REXX program and command responses can be received using a GETMSG functions.

New REXX external functions allow, for example, to respond to a console message, to define an operator communication exit, or to use the VSE lock manager using REXX.

A ready-to-run example is delivered within the REXX/VSE library that implements a console message monitoring system using a Message-Action Table.

Customizing a product must be possible on a native VSE system without relying on the presence of VM or any other development constellation.

Please note that jobs for all these tasks should be supplied as library members of type Z with the product.

More information

Refer to the following IBM manuals for detailed information:

- *VSE/ESA Installation*
- *VSE/ESA System Control Statements*
- *VSE/ESA Guide to System Functions*
- *REXX/VSE User's Guide*
- *REXX/VSE Reference*

Chapter 9. Providing Service

A product's quality depends also on the way it is serviced in the field. The service process is a method to enhance a product's quality and image. This chapter shows how you can provide service for your product using MSHP.

Service can be either corrective or preventive. Corrective service is supplied in response to a defect discovered by an user of the product. Preventive service means applying available service to a product before a defect is rediscovered by another user.

In Chapter 7, “Creating Installation Tapes” on page 79 you saw that each product is structured into components and that each component is identified by its component ID (see “Component Identifier” on page 72) and CLC. Service is always for the component of the product, not for the product itself. Therefore, the fully-qualified component identifier is used to relate the fix to that defective component.

Note

The identifiers used for product installation can't be changed by service.

Figure 8 on page 83 shows that for MSHP a component consists of PHASES, MODULES, and MACROS. Only these kinds of members can be serviced by MSHP. The MACRO member types allocated for specific use are shown in Chapter 11, “Library Member Types” on page 131.

Corrective Service

There are two ways to supply corrective service for VSE products

- Through an APAR fix (ZAP)
- Through a Program Temporary Fix (PTF)

APAR

At IBM, every previously unknown technical problem of a current release of a product is reported in the RETAIN system by an APAR record and uniquely identified by an **APAR number**. APAR stands for Authorized Program Analysis Report.

APAR Number

An APAR number is seven characters long. The first and second character (APAR prefix) is alphabetic; the following five characters are numeric.

For an IBM product the APAR prefix is assigned according to PDR 205 when the component(s) of this product is(are) defined to RETAIN. For independent vendor products, any characters can be chosen for the prefix. However, it is recommended that you don't use prefixes that are already assigned to IBM VSE products (AP, DY, HB, IR PL, PN, PP) or to IBM Business Partner products (SO) to avoid possible confusion for customers.

For information on how the customer can submit an APAR to IBM, refer to "Submitting an APAR" in the *VSE/ESA Guide to Solving Problems* Please remember to set up a contact where your customers can report problems and submit the equivalent of an APAR.

APAR Fix

Code defects reported by an APAR are solved by an **APAR fix**, which is identified by the corresponding APAR number in RETAIN.

The APAR fix is a fast, temporary fix to the problem recorded by the APAR that may be used before the PTF is generally available and is also known as ZAP. Its goal is to give immediate help to a customer to continue work; the fix does not have to be the permanent solution applied to the product later on, as this will be the PTF. Incidentally, ZAP is not an acronym or an abbreviation; the word reflects the speed of the fix.

An APAR fix is available to all users who have encountered that problem. For problems or code changes affecting only one customer for which no APAR was written, a so-called **local fix** is applied. The local fix is written in the same way as the APAR fix.

For a sample of how to write such a fix, see Chapter 12, “APAR Fix (ZAP)” on page 133.

Resolving an APAR via PTF

Normally IBM makes APAR fixes generally available by incorporating them into a PTF.

A PTF is a temporary but **universal** fix resulting from a technical problem in a current release of the program. PTFs are built for all users of a product as a response to a program's error reported in the form of an APAR.

PTFs are the preferred method of service. PTFs allow extensive changes of a product without making it necessary to replace the entire product and requiring reinstallation of a product. Also, one PTF can resolve several APARs. A PTF replaces one or more programming elements: a macro, module, or phase in the component of an installed product. Note that other types of library members, such as procedures (PROC) are **not** supported.

In contrast to the APAR fix, the PTF is not a fast fix but requires changes in the source code. The PTF is temporary only in that the program fix may be designed and/or implemented differently in the next release. The fixes for all problems reported in the form of APARS will be integrated in the next release of the product.

PTF Number

PTFs are identified by a **PTF number**. A PTF number is seven characters long. The first and second character are alphabetic; the following five characters are numeric.

For an IBM product the PTF prefix is assigned according to PDR 205 when the component(s) of this product is(are) defined to RETAIN. For independent vendor products, any characters can be chosen for the prefix. However, it is recommended that you don't use prefixes that are already assigned to IBM VSE products (UD, UG, UL, UN UP, UR) or to IBM Business Partner products (UU) to avoid possible confusion for customers.

Ensuring the Correct Environment

When servicing a product, you need to ensure that the PTF/APAR is applied properly and does not damage the product. A product can be damaged when service is applied to a component on the wrong release level or when prerequisite/corequisite changes (changes that depend on each other) for other components are missing.

Since PTFs/APARs belong to a certain component, MSHP statements ensure that the PTF/APAR is applied to the correct component in the correct environment.

The MSHP statements **APPLY** in a PTF and **CORRECT** in an APAR contain the full qualified component ID. This relates the PTF/APAR to the component and thus ensures the correct target library.

The MSHP statement REQUIRES is used to ensure that the environment is correct. This statement requests that other products, components, PTFs, or APAR fixes should be present, or explicitly not present when the APAR/PTF is installed.

Note

The environment is defined by the identifiers used for product installation (component ID, CLC, and product ID). These identifiers can't be changed by service.

Using REQUIRES: The following example describes a scenario where a technical problem is discovered in an operational environment:

The product to be serviced in this scenario consists of two components, components A and B, each with three modules, as illustrated in Figure 19 on page 104.

CASE 1: Defect in module A1 and A2 of component A

After the new product is shipped, you discover that a defect in component A needs to be fixed in the modules A1 and A2 by means of a PTF.

Solution: PTF UD00001 will replace module A1 and A2 and ensures its correct target by using the appropriate APPLY abd REQUIRES statements:

```
APPLY 5666-001-01-A10:UD00001
REQUIRES PRE=001A10
```

CASE 2: Defect in module A1 that requires also a change in module B1

Some time later, another defect is discovered. Again, a module must be fixed in component A, but this time a change in component B is also required.

Solution: Two PTFs, UD00002 and UD00003, are required, because two components are affected.

PTF UD00002 for component A:

The PTF requires the following for installation: because module A1 now contains the fix for the first problem, which affects also module A2, the application of PTF UD00001 is required. To ensure the application of the earlier PTF, this PTF is stated as a **prerequisite PTF**, as shown below.

Since the new change in module A1 requires the modified module B1 in component B, PTF UD00003 for component B must be applied along with PTF UD00002. This is called a **corequisite PTF**.

The MSHP statements to ensure correct application look as follows:

```
APPLY 5666-001-01-A10:UD00002
REQUIRES PRE=001A10
REQUIRES PRE=UD00001
REQUIRES 5666-001-02 CO=UD00003
```

PTF UD00003 for component B:

The fix in component B replaces module B1 and requires PTF UD00002 for component A to be present, too, so that they are applied together. The MSHP statements to ensure correct application look like this:

```
APPLY 5666-001-02-A10:UD00003
REQUIRES PRE=001A10
REQUIRES 5666-001-01 CO=UD00002
```

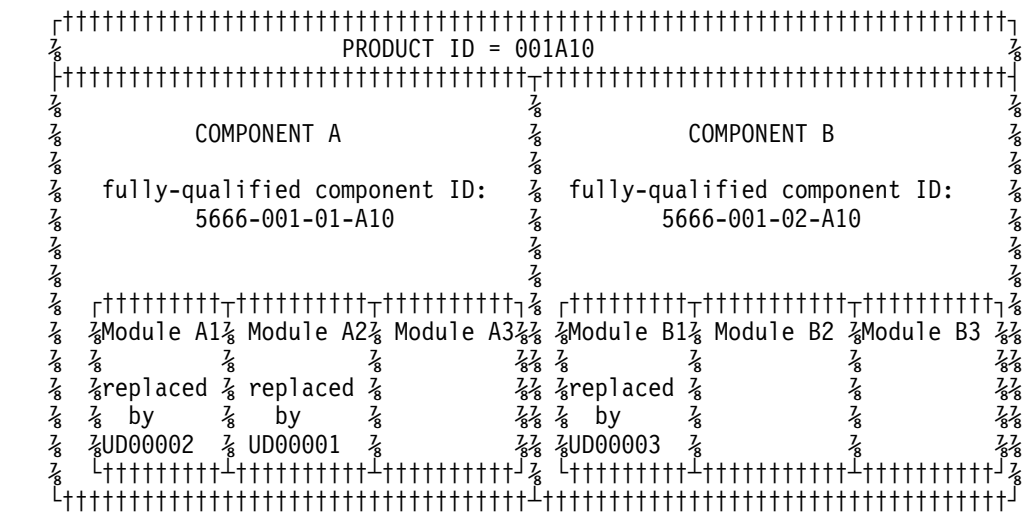


Figure 19. Example of a Serviced Product

Summary: The installation of PTFs and the resolution of prereq/coreq PTF situations are controlled by MSHP as follows:

Prerequisite: One PTF requires the application of another as a prerequisite to its own application.

Corequisite: Two PTFs require the application of each other. The PTFs must be installed together.

For additional sample PTFs refer to Chapter 13, “Programming Temporary Fix (PTF)” on page 135.

For detailed information on JCL and MSHP statements and their additional parameters, please refer to *VSE/ESA System Control Statements*

Building a PTF

Using Figure 20 on page 106 as an example of a PTF, this section shows how a complete PTF can be built.

A PTF consists of three parts:

- JCL with comment cards between // JOB and // EXEC MSHP
- MSHP statements
- The data

Part 1 : **JCL with comment cards between // JOB and // EXEC MSHP**: This comment describes the problem(s) that is/are fixed by this PTF, and whatever else is necessary for successful PTF application. For a sample of a PTF with special instructions in the comment, refer to “PTF for Macros” on page 137.

Part 2 : The **MSHP statements** make up the definition of the PTF. In addition to the environment description, they define which PTFs are superseded, which library members are affected, and which link books have to be relinked.

APPLY is the first MSHP statement in a PTF. It relates the PTF to the component and thus ensures the correct target library. All of the following MSHP statements are related Detail Control Statements.

RESOLVES is a mandatory statement; it indicates which APARs are fixed by the PTF and may associate a comment with a PTF.

REQUIRES is an optional statement and ensures the correct PTF environment. For more examples of REQUIRES, refer to “PTF for Synchronizing Service” on page 138 and the following pages.

SUPERSEDES is an optional statement and identifies which PTFs are superseded by this one. For detailed information, refer to “Superseding PTF and Associated Requires-Groups” on page 139.

AFFECTS is a mandatory statement and lists the phases, modules, and macros that are replaced by the PTF. Multiple PHASES, MODULES, and MACROS operands can be specified. If different macro types are to be serviced in one PTF, at least one MACROS operand must be specified for each macro type.

INVOLVES is an optional statement and is needed if modules are replaced, it must be link-edited after installation. The specified value is the link-book(s).

Note: A link-book must have a member type of OBJ and must not include comments.

DATA is a mandatory statement; it indicates the last MSHP statement in a PTF and is followed by the actual library members that will replace the defective ones.

Part 3 : **Data**: The last PTF part, data, is actually part of the MSHP DATA statement, the new library members. These members (phases, modules, and/or macros) can be listed in any order, however it is recommended to group them. MSHP starts either the linkage editor if it finds a PHASE statement or the librarian program if it finds a CATALOG statement and passes all following lines unaltered and unchecked to the called program. If this program recognizes the end of a member, MSHP gets control back and again starts checking the input lines.

The new modules, phases, or macros are prepared in a VSE library and are punched by using LIBR to build the PTF elements. When punched by LIBR, some necessary statements are supplied by the librarian: the PHASE and/or CATALOG statements, as well as the /+ delimiter statements. They should **not** be removed from the library members. They are used for separation.

```
// JOB UD12345
*  APPLICATION COMMENT(S) :
*                            COMPONENT   :     5666-301-01-A42
*                            APARS FIXED :     DY28888
*
*  SPECIAL CONDITIONS     : NONE
*
*  COPYRIGHT :               (C) COPYRIGHT IBM CORP. 1984
*                            LICENSED MATERIAL - PROPERTY OF IBM
*
*  COMMENTS  :
*
// PAUSE EOB OR CANCEL
// EXEC MSHP
 APPLY  5666-301-01-A42 : UD12345
  RESOLVES 'short description' APARS=DY28888
  REQUIRES PRE=(302H01)
  SUPERSEDES (UD23456,UD34567)
  AFFECTS                   -
  PHASES =  (  PHAABC       -
               PHADEF  )    -
  MODULES  = ( MODGHI       -
               MODJKL  )    -
  MACROS   = ( MACMNO       -
               MACPQR  )    -
  TYPE     =   A            -
  MACROS   =   MACSTU       -
  TYPE     =   E
  INVOLVES LINK = LINKBOOK
  DATA
 PHASE PHAABC,S+X'.......'
 .................PHASE PHAABC
 PHASE PHADEF,S+X'.......'
 .................PHASE PHADEF
 CATALOG MODGHI.OBJ    REPLACE=YES
 .................MODULE MODGHI
 CATALOG MODJKL.OBJ    REPLACE=YES
 .................MODULE MODJKL
 CATALOG MACMNO.A      REPLACE=YES
 .................MACRO  MACMNO
 CATALOG MACPQR.A      REPLACE=YES
 .................MACRO  MACPQR
 CATALOG MACPQR.E      REPLACE=YES
 .................MACRO  MACSTU
/$
/*
/&
```

Figure 20. PTF Format

Distributing a PTF

PTFs are usually shipped on a tape called **VSE service tape**. A VSE Service Tape can contain only one PTF, but can also consist of up to 9 tape volumes. It has the following layout:

Figure 21. Layout of PTF Tape	
File Number	**Content**
1	History file prepared by MSHP 'LIST SERVICETAPE XREF'
2	Service documentation
3	null file (tape mark)
4	EXCLUDE-list
5	Cover letters
6	PTFs
7	null file (tape mark)
8	null file (tape mark)

Description: All files that are relevant to MSHP (file 1, 2, 4, 5, and 6) must either contain MSHP information in the correct format or must be empty. File 2 has to be blocked with the block size of 7980; the logical record size has to be 133 characters (fixed format). File 4, file 5, and file 6 have to be blocked with the block size of 10320; the logical record size has to be 80 characters (fixed format).

The files used by MSHP are described below; the files not used by MSHP are present for compatibility reasons (pre-VSE/SP 2.1 format) only and may be empty.

File 1: TAPE HISTORY

This file is created as follows:

- Create tape with 5 tape marks and all PTFs on file 6.
- EXEC MSHP with LIST SERVICETAPE XREF.
- BACKUP history auxiliary to a work tape.
- Copy auxiliary history to file 1 of the final service tape.

The history on file 1 contains all MSHP information about the PTFs in file 6. It improves the performance of the PTF application process if it exists; it can, however, be empty (null file).

File 2: Service Documentation

May contain any information the customer should get together with a VSE Service Tape; it can, however, be empty (null file). Information in file 2 can be printed to SYSLST using the option Print Service Documentation of the PTF Handling dialog or the MSHP LIST function.

File 4: EXCLUDE-list

Contains any sequence of EXCLUDE control statements, indicating which PTFs, components, or products have to be automatically excluded from the service application process unless they are explicitly requested by the user via the INCLUDE command. The control statements have to be specified according to the syntax of the MSHP command EXCLUDE. Examples for FILE 4:

```
EXCLUDE PTF=(UD00001, UD00017, UD12345);

EXCLUDE PTF=(UD98765);

EXCLUDE COMPONENT=5686-066-07-15C
```

The file may be empty (null file) if nothing is to be excluded.

File 5: Cover letters

This file contains any sequence of the cover letters of the PTFs in file 6. It can be used to optimize the LIST SERVICE COVER function of MSHP, which produces a list of cover letters of all PTFs on the service tape. If this file is empty (null file), the cover letters are selected from the PTFs in file 6.

A cover letter describes the PTF. It consists of the first two parts of a PTF as described under “Building a PTF” on page 104, that is, a cover letter is a PTF without the data.

For a sample of a cover letter containing special PTF instructions, refer to “PTF for Macros” on page 137.

FILE 6: PTFs

Contains any sequence of PTF jobs built as described under “Building a PTF” on page 104.

Installing a PTF

PTF installation on VSE is done by MSHP. MSHP can be used either natively or with the support of the service dialogs provided by the VSE/SP Unique Code Interactive Interface (II). The recommended way is to use the dialogs.

VSE/SP Unique Code Dialogs: To assist the customer in using MSHP, the VSE/SP Unique Code II. provides the following dialog support for service application:

Apply PTF from Service Tape

This dialog is mainly used for PTF mass application and merges all applicable PTFs from a service tape directly into the product (sub)libraries of a running system. However, individual PTFs can be selected for installation with the INCLUDE option or excluded from installation with the EXCLUDE option.

Analyze and Apply Service Tape

This dialog maintains a list of service tapes that have been analyzed and allows to display various information about the PTFs on these tapes. The dialog provides:

- List functions based on tape, product, component, or PTF.
- Information on:
  - Affected sub-library(ies)
  - Affected members
  - Requisites
  - APARs fixed
- Select functions to choose service based on product, component, or PTF level for products that are installed.

Note: Although this dialog displays the requisites of the listed PTFs, it does not check whether all requirements are met. It only checks whether a listed PTF is already installed. Complete requisite checking is done by MSHP.

Both dialogs support **Indirect Service Application** for system libraries. That means, service is not applied directly to these libraries, but to additional temporary libraries. This allows to test the installed service before applying it to the system libraries. Indirect service application is either defined by the PTF's APPLY statement or may be specified by the customer using the Force Indirect option.

MSHP Processing Sequence: MSHP is able to service as many components as requested in one step and one run through the service tape(s), if the system history file correctly reflects the sub-libraries in which the components reside, and provided those sub-libraries are on-line.

Based on the history information and using the PTF's MSHP control statements, MSHP is able to determine:

- Which PTFs from a service tape have to be applied to which product in which library.
- Which requisites are needed.
- Whether service may conflict with an already applied APAR fix or local fix.

MSHP installs PTFs by replacing existing members in the VSE library with the updated members provided by PTFs.

Note

New members can be added, but existing members cannot be deleted.

The necessary steps to activate the correction in the system are described in the PTF coverletter and in the JCL comments of the apply job.

Before applying PTFs, MSHP builds a temporary history file, where the information about all requested PTFs is gathered. It uses either the tape history file (file 1), if available, or it picks up all PTF information from file 6, the PTF file. MSHP determines which PTFs are to be applied: those that are requested by the customer and those that are needed to meet the requirements of the requested ones. For the elected ones, MSHP checks the PTF requirements and protects local APAR fixes and user generated members, if both, PTF and system history information, are correct. If more than one PTF affects the same member, MSHP determines (based on the requisite relationship) which PTF contains the latest level of this member. At application time, only this member is picked up, the older levels are ignored.

PTF data has not been moved so far. A cross-reference list of all applicable PTFs and APARs is then printed and the user is asked for confirmation, before MSHP starts to pick up the affected members from the tape, to replace them directly in the user's sub-libraries and to update the history file.

Revoking PTFs and APARs

Service that is installed REVOKABLE can be removed if deemed necessary.

APAR fixes and local fixes are installed with a default option of REVOKABLE in the CORRECT statement. For PHASEs and MODULEs, MSHP stores all relevant information in the system history file. These APARs can be removed with the UNDO function. For source members, a job is punched to SYSPCH, which can be started to restore the original source member, thus removing all changes done by the APAR fix.

Single PTFs application (via SYSRDR/SYSIN): If PTFs are installed using the REVOKABLE option of the APPLY statement, backout jobs are created on tape, which can be run to restore the original members.

Mass Application of PTFs: If PTFs are installed from a service file using the INSTALL SERVICE command, and the REVOKABLE option is specified, a BACKOUT job is created for each component and written to a backout tape. The entire service can be revoked by executing the INSTALL BACKOUT command, should this be necessary.

Note: Since MSHP selects only the members at the highest service level for installation, either **ALL** PTFs of a MSHP service application job can be revoked or none.

Preventive Service

There are two methods to offer preventive service to customers: as cumulative service tape or as a product refresh.

Cumulative Service Tape

All PTFs that have assembled in the course of time are merged and put on one tape, the cumulative service tape (also known as Program Update Tape). This tape is shipped to the customer.

Refresh

This method is used for VSE packages and its optional products. The product is upgraded with all available service (PTFs), tested and then sent to the distribution centers. A product refresh ensures that new customers will receive the product with most of the available service already installed and current customers can order a system refresh (free of charge) to upgrade the existing system.

VSE Refresh Installation

A VSE package refresh consists of updated base product libraries and of updated optional products. The VSE base system refresh can be installed via the **Fast Service Upgrade (FSU)**process, optional products have to be re-installed after the FSU process.

Since a FSU replaces complete libraries, a system upgrade with FSU can be much faster (depending on number of PTFs) than installing the same amount of PTFs with MSHP, which replaces single members.

More information

Refer to the following IBM manuals for detailed information:

- *VSE/ESA System Control Statements* for information on how PTFs are built.
- *VSE/ESA Guide to Solving Problems* for information on how to submit an APAR.
- *VSE/ESA Installation* for information on PTF handling and running a FSU in VSE/ESA Version 1.
- *VSE/ESA System Upgrade and Service* for information on PTF handling and running a FSU in VSE/ESA Version 2.

Part 5. Packaging and Service Samples

This part provides primarily packaging and service samples. How to create installation tapes and how to service your product is described in Part 4, “Creating Installation Tapes and Servicing Your Product” on page 69.

Chapter 10. Packaging of Products

Products are packaged in VSE/Advanced Functions Version 2 format. The following jobs are examples only, and should be adapted to the available installation and completed by adding the necessary LIBDEF statements.

Note: To shorten the examples shown, parts were extracted and replaced by periods.

Library Creation

Creation of a Library on a Sequential Disk Extent

There is a sample skeleton like the one shown in Figure 22 in ICCF library 59 called SKLIBDEF.

```
// JOB CREATE LIBRARY         /* CREATE LIBRARY FOR PROGRAM PRODUCT  */
// OPTION PARSTD=ADD          /* OPTION PARTITION STANDARD LABEL     */
// ASSGN SYS007,140           /* ASSIGN DISK IF NECESSARY            */
// DLBL PRODLIB,'NEW.PRIVATE.LIB'       /* LABELS AND EXTENTS FOR    */
// EXTENT SYS007,VOLID5,1,0,190,114    /*PRIVATE LIBRARIES           */
// EXEC LIBR                  /* EXECUTE LIBRARIAN PROGRAM           */
   DEFINE LIB=PRODLIB         /* DEFINE THE LIBRARY 'PRODLIB' FOR    */
/*                            /* LATER INSTALLATION STEP             */
/&
```

Figure 22. Creating a Library on a Sequential Disk Extent

Definition of a Library in VSAM Managed Space

The VSE dialogs should be used to define a library in VSAM-managed space. The following job sequence can also be used:

```
// JOB DEFINE
*  DEFINE MASTER/USER CATALOG, SPACE, CLUSTER
// OPTION STDLABEL=ADD
// DLBL  IJSYSCT,'VSAM.MASTER.CATALOG',,VSAM
// EXTENT SYSCAT,SYSRES,1,0,4465,95
// DLBL DNAME,'USER.CATALOG.NO1',,VSAM
// EXTENT SYS003,DOSWK1,1,0,4465,95
// DLBL SPACE,'VSAM.DATA.SPACE',,VSAM
// EXTENT SYS004,DOSWK2,1,0,10,1900
// ASSGN SYS003,cuu
// ASSGN SYS004,cuu
/*
// EXEC IDCAMS,SIZE=AUTO
   DEFINE MASTERCATALOG               -
          (NAME(VSAM.MASTER.CATALOG)  -
          VOLUME(SYSRES)              -
          CYL(5 0))
   DEFINE USERCATALOG                 -
          (NAME(USER.CATALOG.NO1)     -
          VOLUME(DOSWK1)              -
          CYL(5  0))                  -
          CATALOG(VSAM.MASTER.CATALOG)
   DEFINE SPACE                       -
          (FILE(SPACE)                -
          CYL(100 0)                  -
          VOL(DOSWK2))                -
          CATALOG(USER.CATALOG.NO1)
   DEFINE CLUSTER                     -
          (NAME(VSE.PRODUCT.LIBRARY)  -
          NONINDEXED                  -
          SHAREOPTION(3)              -
          RECORDFORMAT(NOCIFORMAT)    -
          CYL(20 10))                 -
          CATALOG(USER.CATALOG.NO1)
/*
/&
// JOB DEFINEL
*  DEFINE LIBRARY IN VSAM SPACE
// DLBL PRODLIB,'VSE.PRODUCT.LIBRARY',,VSAM,DISP=(OLD,KEEP)
// EXEC LIBR
   DEFINE LIB=PRODLIB
/*
/&
```

Figure 23. Defining a Library in VSAM Managed Space

Creating the Header

Cataloging a File that will Become the Header

```
* $$ JOB JNM=CATHDR,CLASS=0,DISP=D
// JOB CATHDR  CATALOG THE Z BOOK THAT CONTAINS THE COPYRIGHT INFO
// DLBL lib,'product.library.name',99/365,SD
// EXTENT ,vvvvvv,1,0,sss,nnnn
// EXEC LIBR
 ACC S=lib.sublib
 CATALOG member.Z  R=Y
insert header information    /* the info here is used at a later time
 . see sample below          /* by MSHP to create the tape header
 .                           /* copyright information
/+
/*
/&
* $$ EOJ
```

Figure 24. Cataloging a File that will Become the Header

Header Information

Product Containing "Restricted Materials" (non-OCO product)

```
HDR00101  LICENSED MATERIALS - PROPERTY OF IBM
HDR00102  THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM"
HDR00103  <pgm-nr> (C) COPYRIGHT IBM CORP. 19xx, 19yy.
HDR00104  ALL RIGHTS RESERVED.
HDR00105  US GOVERNMENT USERS RESTRICTED RIGHTS -
HDR00106  USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
HDR00107  GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
HDR00108  SEE COPYRIGHT INSTRUCTIONS.
HDR09901  END
```

Figure 25. Header for Product Containing "Restricted Material"

Product not Containing any "Restricted Materials" (OCO product)

```
HDR00101  LICENSED MATERIALS - PROPERTY OF IBM
HDR00102  <pgm-nr> (C) COPYRIGHT IBM CORP. 19xx, 19yy.
HDR00103  ALL RIGHTS RESERVED.
HDR00104  US GOVERNMENT USERS RESTRICTED RIGHTS -
HDR00105  USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
HDR00106  GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
HDR09901  END
```

Figure 26. Header for Product not Containing "Restricted Material"

Note

1. IBM developers should check Corporate Standard C-S 0-6045-002 for the latest wording of the copyright statement
2. 19xx first year product was published
3. 19yy last year product was published with substantial changes (5% or more changes from the original)

For an OCO Product with More than One Copyright Information

```
HDR00101 LICENSED MATERIALS - PROPERTY OF IBM
HDR00102 5686-022 (C) COPYRIGHT IBM CORPORATION  1990
HDR00103 ALL RIGHTS RESERVED.
HDR00104 US GOVERNMENT USERS RESTRICTED RIGHTS -
HDR00105 USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
HDR00106 GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
HDR00107
HDR00108 ENGLISH SYNONYM INFORMATION IS BASED ON THE AMERICAN HERITAGE
HDR00109 DICTIONARY DATA BASE - ROGET'S II, THE NEW THESAURUS - OWNED
HDR00110 BY HOUGHTON MIFFLIN COMPANY AND USED WITH PERMISSION.
HDR00111
HDR00112 (C) COPYRIGHT HOUGHTON MIFFLIN COMPANY 1982
HDR00113
HDR00114 GRADE DATA IS USED WITH PERMISSION FROM THE LIVING WORD
HDR00115 VOCABULARY.
HDR00116
HDR00117 (C) COPYRIGHT WORLD BOOK, INC. 1984
HDR00118
HDR00119 SPANISH SYNONYM INFORMATION IN BASED ON THE DICCIONARIO
HDR00120 ESPANOL DE SINONIMOS Y ANTONIMUS, 8TH EDITION (ELEVENTH RE-
HDR00121 PRINT), PUBLISHED BY AGUILAR S.A. IN MADRID, SPAIN AND USED
HDR00122 WITH PERMISSION.
HDR00123
HDR00124 (C) COPYRIGHT FEDERICO CARLOS SAINZ DE ROBLES 1984
HDR00125
HDR00126 SWEDISH SYNONYM INFORMATION IS BASED ON THE STORA
HDR00127 SYNONYMORDBOKEN, PUBLISHED BY STROMBERG'S IN STOCKHOLM,
HDR00128 SWEDEN, AND USED WITH PERMISSION.
HDR00129
HDR00130 (C) COPYRIGHT ALVA STROMBERG 1952
HDR00131
HDR09901 END
```

Figure 27. Header for an OCO Product with More than One Copyright Information

Creating or Changing VSE/Advanced Functions History Information

Creating History Information

History Information for One Component

Information using MSHP: In this sample, the extent information for the product's history file is supplied by the MSHP detailed statement DEFINE.

```
// JOB DW210BA
* DEFINE HISTORY FOR DW/370 2.1.0
// ASSGN SYS021,DISK,VOL=DW202T,SHR
// EXEC MSHP,SIZE=1024K
  CREATE HISTORY SYSTEM
    DEFINE HISTORY SYSTEM UNIT=SYS021     -
                       EXTENT=0705:08     -
          ID='DW/370 5686-022 2.1.0 BASIC.HISTORY'
  ARCHIVE  5686-022-01-A10           /* FULLY QUALIFIED COMPONENT-ID*/
  ARCHIVE  022A10                    /* PRODUCT ID                  */
    RESOLVES 'DW/370BASE.2.1.0 -  5686-022'
    COMPRISE 5686-022-01                                  -
         PHASES=(DDD*,DKL*,EDD*,EKL*,T1D*,$$$CO*)         -
         MODULES=(DDD*,DKL*,EDD*,EKL*,$$$CO*)             -
         MACROS=(DDD*,EDD*,$$$CO*)                        -
           TYPE=A
    COMPRISE 5686-022-01                                  -
         MACROS=DDD*                                      -
           TYPE=E
    COMPRISE 5686-022-01                                  -
         MACROS=(DDD*,HD*)                                -
           TYPE=Z
    COMPRISE 5686-022-01                                  -
         MACROS=(CLI*,DW*,EMPT*,HEX,DKL*,EX*)             -
           TYPE=X
  RESIDENCE PRODUCT=022A10           -
         PRODUCTION=DW202DA.PR$A10
/*
/&
```

Figure 28. Extent Information for the History File Using MSHP

Information using Job Control: In this sample, the extent information for the product's history file is supplied by JOB CONTROL statements.

```
// JOB VSAM BASE HISTORY
// DLBL IJSYSHF,'VSM.H21.HISTORY.FILE' /* DEFINE HISTORY FOR VSAM BASE */
// EXTENT SYS012,,1,0,7550,10        /* ON TRACK 7550 AND SIZE OF 10   */
// ASSGN SYS012,130                  /* ASSIGNMENT FOR THE HISTORY     */
// EXEC MSHP,SIZE=1024K
ARCHIVE 5686-066-05-15C
ARCHIVE 06615C
RESOLVES 'VSE/VSAM VERSION 6.1.0'
COMPRISES 5686-066-05 -
PHASES = ( $$$COC66 -
           $$BACLOS -
           $SVAVSAM -
           . . .    -
           IDC*     -
           IIP*     -
           IKQ*   ) -
MACROS = ( BLDVRP   -
           DLVRP    -
           ENDREQ   -
           ERASE    -
           . . .    -
           SHOWCAT  -
           TCLOSE   -
           WRTBFR ) TYPE=A ;
COMPRISES 5686-066-05 -
MACROS = ( HD06615C) TYPE=Z ;
RESIDENCE PRODUCT=06615C PRODUCTION=PRODPPS.PRVSM610 ;
/*
/&
```

Figure 29. Extent Information for the History File Using Job Control

History Information for a Feature of a Product: If you compare this sample to the job of the base product on the previous page, you will notice that the history information for a feature of a product requires only this one additional statement:

```
REQUIRES PRE=06615C         /* PRE-REQUISITE BASIC PRODUCT */
```

```
// JOB VCM610 HISTORY FILE
// DLBL IJSYSHF,'VCM.H01.HISTORY.FILE' /* DEFINE HIST. FOR VSAM FEAT.*/
// EXTENT SYS012,,1,0,7560,10          /* ON TRACK 7560, SIZE OF 10  */
// ASSGN SYS012,130                    /* ASSIGNMENT FOR HIST.FILE   */
// EXEC MSHP,SIZE=1024K
ARCHIVE 5686-066-03-15G
ARCHIVE 06615G
RESOLVES 'VSE/VSAM VERSION 6.1.0 COMMON MACROS'
REQUIRES PRE=06615C ;                  /* VSE/VSAM VERSION 6.1.0     */
COMPRISES 5686-066-03 -
MACROS = ( ACB      -
           EXLST    -
           . . .    -
           SHOWCB   -
           TESTCB   ) TYPE=A ;
COMPRISES 5686-066-03 -
MACROS = ( HD06615G) TYPE=Z ;
RESIDENCE PRODUCT=06615G PRODUCTION=PRODPPS.PRVCM610 ;
/*
/&
```

Figure 30. Creating History Information for a Feature of a Product

History Information for a Product Consisting of Multiple Components Installed Together

```
// JOB VSECF61 HISTORY FILE
// DLBL IJSYSHF,'VSE/AF.NEW.HIST',99/365,SD
// EXTENT SYS012,SYS380,1,0,1560,45
// ASSGN SYS012,380
// EXEC MSHP,SIZE=1024K
CREATE HIST SYSTEM
ARCHIVE 5686-066-01-15C ;
ARCHIVE 5686-066-02-15C ;
ARCHIVE 5686-066-03-15C ;
ARCHIVE 5686-066-04-15C ;
ARCHIVE 5686-066-05-15C ;
ARCHIVE 5686-066-06-15C ;
ARCHIVE 5686-066-07-15C ;
ARCHIVE 5686-066-08-15C ;
ARCHIVE 5686-066-09-15C ;
ARCHIVE 06615C          /* LEVEL ID FOR VSE/CF VERSION 6.1.0 */ ;
RESOLVES 'VSE CENTRAL FUNCTIONS VERSION 6.1.0' ;
COMPRISES 5686-066-02 -                       /* OK */
PHASES   = ( $$ABERRF  -
             $$ABERRG  -
             $$ABERRK  -
             $$ABERRL  -
             $$ABERRM  -
             $$ABERRN  -
             $$ABERRO  -
             . . .     -
             IJDANCHX  -
             IJDPR3  -
             IJH*    -
             IJJ*     )  ;
```

Figure 31 (Part 1 of 3). History Information for a Product with Multiple Components Installed Together

```
COMPRISES 5686-066-02 -
MODULES  = ( $$ABEREF  -
             $$ABERRG  -
             $$ABERRK  -
             . . .     -
             IJJGSD*   -
             IJJGV*    -
             IJJH*     -
             IJJT*     -
             IJJX*     -
             IJND*   ) ;
COMPRISES 5686-066-02 -                       /* OK */
MACROS   = ( CDMOD     -
             CHECK     -
             . . .     -
             . . .     -
             TRUNC     -
             WRITE  )     TYPE=E ;
COMPRISES 5686-066-02 -                       /* OK */
MACROS   = ( BOMTAC    -
             IJJT$SEC )   TYPE=A ;
COMPRISES 5686-066-04 -                       /* OK */
PHASES   = ( $$ABERAN -
             $$ABERRS -
             . . .    -
             $$BOOR01 ) -
MODULES  = ( $$ABERAN -
             $$ABERRS -
             . . .    -
             $$BOMR01 -
             $$BOOR01 ) -
MACROS   = ( DFR       -
             DISEN     -
             . . .     -
             RESCN     -
             SETDEV     ) TYPE=A ;
```

Figure 31 (Part 2 of 3). History Information for a Product with Multiple Components Installed Together

```
COMPRISES 5686-066-06  -
PHASES   = ( $$$CO15C -
             $$A$*    -
             $$ABERA1 -
             . . .    -
             SAFORMEM -
             SSERV      ) ;
COMPRISES 5686-066-06    -                    /* OK */
MODULES  = ( $$$CO15C -
             $$A$IPL* -
             . . .    -
             VMCF$BCP -
             VMCFCP     ) ;
COMPRISES 5686-066-06   -
MACROS   = ( APL      -
             APPCVM   -
             . . .    -
             MSAT     -
             MVCOM     ) TYPE=A ;
COMPRISES 5686-066-06  -
MACROS   = ( NPGR     -
             NPGRLST  -
             . . .    -
             XPOST    -
             XWAIT     )   TYPE=E ;
COMPRISES 5686-066-06  -
MACROS   = ( HD06615C  )   TYPE=Z ;
COMPRISES 5686-066-07 -                       /* OK */
PHASES   = ( MSHP*      -
             PTF*     ) -
MODULES  = ( IKR*     ) ;
COMPRISES 5686-066-08 -
PHASES   = ( BLN*     -
             BLX*     -
             INFO*    ) -
MODULES  = ( BLN*     -
             IJBXA*   -
             BLX*     ) -
MACROS   = ( BLN*    )   TYPE=T ;
COMPRISES 5686-066-08 -
MACROS   = ( BLN*    )   TYPE=M ;
COMPRISES 5686-066-08 -
MACROS   = ( BLN*    )   TYPE=N ;
RESIDENCE PRODUCT=06615C PRODUCTION=IJSYSR1.SYSLIB -
                         GENERATION=GENLIB1.G1$007 ;
/*
/&
```

Figure 31 (Part 3 of 3). History Information for a Product with Multiple Components Installed Together

History Information for a Product Consisting of Multiple Components Installed Selectively

```
// JOB MUEDHIS
* DEFINE HISTORY FOR DW/370 2.1.0 DICTIONARY
// DLBL DW210A1,'DW370.BAM.LIBRARY.R210'
// EXTENT ,DW210A
// EXTENT ,DW210B
// ASSGN SYS021,DISK,VOL=DW202T,SHR
// EXEC MSHP,SIZE=1024K
 /*          DEFINE HISTORY FOR LEGAL DICTIONARY           */
  CREATE HISTORY SYSTEM
    DEFINE HISTORY SYSTEM UNIT=SYS021     -
                        EXTENT=0935:05     -
          ID='DW/370 5686-022 2.1.0 LEGA.DICT.HISTORY'
  ARCHIVE  5686-022-32-A41           /* COMPONENT-ID AND RELEASE-NO */
  ARCHIVE  022A41                    /* PRODUCT IDENTIFIER          */
    RESOLVES 'DW.DI-LEGA.2.1.0 -  5686-022'
    COMPRISE 5686-022-32                            -
         PHASES=(EKL*)                              -
         MODULES=(EKL*)                             -
         MACROS=($$$CO*)                            -
           TYPE=A
    COMPRISE 5686-022-32                            -
         MACROS=(EKL*)                              -
           TYPE=X
    COMPRISE 5686-022-32                            -
         MACROS=(HD*)                               -
           TYPE=Z
    REQUIRES PRE=022A10              /* PRE-REQUISITE BASIC PRODUCT */
  RESIDENCE PRODUCT=022A41                          -
         PRODUCTION=DW210A1.PR$A41
 /*          DEFINE HISTORY FOR BRAZILIANN DICTIONARY      */
  CREATE HISTORY SYSTEM
    DEFINE HISTORY SYSTEM UNIT=SYS021     -
                        EXTENT=0950:05     -
          ID='DW/370 5686-022 2.1.0 BPOR.DICT.HISTORY'
  ARCHIVE  5686-022-15-A24           /* COMPONENT-ID AND RELEASE-NO */
  ARCHIVE  022A24                    /* PRODUCT IDENTIFIER          */
    RESOLVES 'DW.DI-BPOR.2.1.0 -  5686-022'
    COMPRISE 5686-022-15                            -
         PHASES=(EKL*)                              -
         MODULES=(EKL*)                             -
         MACROS=($$$CO*)                            -
           TYPE=A
    COMPRISE 5686-022-15                            -
         MACROS=(EKL*)                              -
           TYPE=X
    COMPRISE 5686-022-15                            -
         MACROS=(HD*)                               -
           TYPE=Z
    REQUIRES PRE=022A10              /* PRE-REQUISITE BASIC PRODUCT */
  RESIDENCE PRODUCT=022A24                          -
         PRODUCTION=DW210A1.PR$A24
/&
```

Figure 32. History Information for a Product Consisting of Multiple Components Installed Selectively

Adding, Changing and Restoring History Information

Adding Information to an Existing History File: This job removes the default product ID generated by MSHP during the installation of the product on the VSE/ESA system. The product ID is then re-archived to permit the addition of the new RESOLVES statement that contains the 16 character tapefile ID. The sublibrary in the RESIDENCE statement should use the name of the sublibrary that contains the product.

```
* $$ JOB JNM=NEWHIST,DISP=D,CLASS=0
// JOB NEWHIST  CORRECT THE HISTORY FILE
// DLBL IJSYSHF,'history.filename',99/365,SD
// EXTENT SYSxxx,vvvvvv,1,0,sss,nnnn
// ASSGN SYSxxx,DISK,VOL=vvvvvv,SHR
// EXEC MSHP,SIZE=1024K
  REMOVE    prod-id
  ARCHIVE   prod-id
  RESOLVES 'tape.file.id ord.number product release'
  COMPRISES    component ID        -
    PHASES  = (generic names,...)  -
    MODULES = (generic names,...)  -
    MACROS  = (generic names,...) TYPE=A
  RESIDENCE PRODUCT=prod-id PRODUCTION=lib.sublib
/*
/&
* $$ EOJ
```

Figure 33. Adding Information to an Existing History File

Removing Product and Component Identifiers: When removing product and component identifiers, you can either include both REMOVE statements in one job, or submit two separate jobs as shown below.

```
// JOB REMOVE PRODID
// EXEC MSHP,SIZE=1024K
REMOVE 09916O
/*
/&
// JOB REMOVE COMP
// EXEC MSHP,SIZE=1024K
REMOVE 5648-099-01-16O
/*
/&
```

Figure 34. Removing Product and Component Identifiers

Changing Entries in an Existing Product History: This job first archives new product information into a new defined history file. With the MERGE step those parts of the product definition that are not defined are taken over from the old product history.

```
// JOB UPDATE HISTORY             /* UPDATE HISTORY FOR PROGRAM PRODUCT  */
// DLBL IJSYSHF,'NEW.H01.HISTORY.FILE' /* DEFINE NEW HISTORY             */
// EXTENT SYS013,,1,0,7550,10          /* ON TRACK 7550 AND SIZE OF 10   */
// ASSGN SYS012,130               /* ASSIGNMENT FOR THE AUX HISTORY      */
// ASSGN SYS013,130               /* ASSIGNMENT FOR THE NEW HISTORY      */
// EXEC MSHP,SIZE=1024K
   CREATE HISTORY SYSTEM          /* CONTROL STATEMENT FOR CREATION      */
   ARCHIVE 5666-302-01-HO1        /* COMPONENT-ID AND LEVEL              */
   ARCHIVE 302HO1                 /* NEW MSHP PRODUCT ID                 */
   RESOLVES 'PROG.PROD..2.1.0'    /* COMMENT FOR RETRACE PRODUCT         */
   REQUIRES PRE=(AM4F98)          /* UPDATE OF THE PREREQUISITES         */
   OR                             /* UPDATE OF THE PREREQUISITES         */
   REQUIRES PRE=(301HO7)          /* UPDATE OF THE PREREQUISITES         */
   COMPRISES 5666-30-201       -  /* COMPONENT THAT IS COMPRISED         */
   PHASES  = (  $$$COIBM       -  /* DEFINITION OF THE PHASES,MODULES,   */
                AAA*           -  /* MACROS CONTAINED IN THIS PROGRAM    */
                BBBBB  )       -  /* PRODUCT.                            */
   MODULES = (  AAA*   )       -
   MACROS  = (  MACAAAAA       -
              MACBBBBB         -
              CCC*             -
              DDD  )  TYPE=A
   MERGE HISTORY               -  /* MERGE OLD SYSTEM ENTRIES TO THE NEW */
   AUXILIARY SYSTEM               /* HISTORY WITH THE UPDATED ENTRIES.   */
   DEFINE HISTORY AUXILIARY    -  /* DETAIL STATEMENT FOR 'OLD' HISTORY  */
   EXTENT=7220:95              -  /* ON TRACK 7220 AND A SIZE OF         */
   ID='HO1.PRD.HISTORY.FILE'   -  /* 95 TRACKS . PROVIDE AN IDENTIFIER   */
   UNIT=SYS012                    /* AND WHERE THE HISTORY RESIDES       */
/*
/&
```

Figure 35. Changing Entries in an Existing Product History

Restoring the History File

```
// JOB RESTORE HISTORY FILE       /* JOB TO RESTORE MSHP HISTORY         */
// ASSGN SYS006,280               /* ASSIGNMENT FOR TAPE UNIT 280        */
// MTC REW,SYS006                 /* REWIND TAPE                         */
// MTC FSF,SYS006,2               /* TAPE POSITIONING AT HISTORY FILE    */
// EXEC MSHP,SIZE=1024K
RESTORE HISTORY SYSTEM            /* MSHP FUNCTION STATEMENT             */
 DEFINE HISTORY SYSTEM         -  /* DETAIL STATEMENT FOR SYSTEM HISTORY */
  EXTENT=7550:10               -  /* ON TRACK 7550 AND A LENGTH OF       */
  ID='SYSTEM.HISTORY.FILE'     -  /* 10 TRACKS . PROVIDE AN IDENTIFIER   */
  UNIT=SYS012                     /* AND WHERE THE HISTORY SHOULD BE     */
/*                                /* RESTORED.                           */
// MTC REW,SYS006                 /* REWIND TAPE
/&
```

Figure 36. Restoring the History File

Backup of a Product or Feature

Back up the Production and Generation Part of a Product

General Format:

```
* $$ JOB  JNM=BCKNEW,CLASS=0,DISP=D
// JOB BCKNEW   BACKUP PRODUCT IN NEW FORMAT
// DLBL lib,'product.library.name',99/365
// EXTENT ,vvvvvv,1,0,sss,nnnn
// DLBL IJSYSHF,'history.file.name',99/365,SD
// EXTENT SYSxxx,vvvvvv,1,0,sss,nnnn
// ASSGN SYSxxx,DISK,VOL=vvvvvv,SHR
// ASSGN SYS006,cuu               /* of the tape unit               */
// MTC REW,SYS006
// MTC WTM,SYS006,2               /* Optional. To ensure that new   */
// MTC REW,SYS006                 /* tapes are initialized.         */
// EXEC MSHP,SIZE=1024K
 BACKUP PROD=(prod-id) ID='tapefile-id' HEADER=prod-id PROD GENE
/*
// MTC REW,SYS006
/&
* $$ EOJ
```

Figure 37. Backing Up a Product or Feature

The distribution tape created through this backup operation now has the format described in Figure 10 on page 86.

Back up the Production Part of a Product or Feature

```
// PAUSE
// JOB BACKUP DITTO/ESA 1.1.0
// ASSGN SYS006,280
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K
BACKUP PRODUCT=099160 ID='DITTO/ESA..1.1.0' HEADER=HD099160 PROD
/*
// MTC REW,SYS006
/&
```

Figure 38. Backing Up the Production Part of a Product

Back up a Product or Feature for Selective Installation

```
// JOB MUETBASE
// ASSGN SYS021,DISK,VOL=DW202T,SHR            /* DW/370 HISTORIES */
// PAUSE TAPE M1669 ON 570 MOUNTED ?  --->  PRESS ENTER, IF MOUNTED.
// ASSGN SYS006,570,D0                          /* SYS006 DISTR. TAPE*/
// MTC REW,570
// EXEC MSHP,SIZE=1024K
   BACKUP  PRODUCT=022A10                     -
           ID='DW/370BASE.2.1.0'              -
           HEADER=HD022A10                    -
           PRODUCTION
     DEF HISTORY SYSTEM                       -
             EXTENT=0705:08  UNIT=SYS021      -
             ID='DW/370 5686-022 2.1.0 BASIC.HISTORY'
/*
// EXEC MSHP,SIZE=1024K
   BACKUP  PRODUCT=022A24                     -
           ID='DW.DI-BPOR.2.1.0'              -
           HEADER=HD022A24                    -
           PRODUCTION
     DEF HISTORY SYSTEM                       -
             EXTENT=0950:05  UNIT=SYS021      -
             ID='DW/370 5686-022 2.1.0 BPOR.DICT.HISTORY'
/*
// EXEC MSHP,SIZE=1024K
   BACKUP  PRODUCT=022A41                     -
           ID='DW.DI-LEGA.2.1.0'              -
           HEADER=HD022A41                    -
           PRODUCTION
     DEF HISTORY SYSTEM                       -
             EXTENT=0935:05  UNIT=SYS021      -
             ID='DW/370 5686-022 2.1.0 LEGA.DICT.HISTORY'
/*
// MTC RUN,SYS006
/&
```

Figure 39. Backing Up a Product or Feature for Selective Installation

Installing a Product or Feature

Installing a Product with a Production Part

```
* $$ JOB JNM=INSNEW,CLASS=0,DISP=D
// JOB INSNEW  INSTALL A NEW FORMATTED TAPE
// DLBL lib,'product.library.name',99/365
// EXTENT ,vvvvvv,1,0,sss,nnnn
// DLBL IJSYSHF,'history.file.name',99/365,SD
// EXTENT SYSxxx,vvvvvv,1,0,sss,nnnn
// ASSGN SYSxxx,DISK,VOL=vvvvvv,SHR
// ASSGN SYS006,cuu                          /* CUU OF TAPE UNIT */
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K
 INSTALL PRODUCT FROMTAPE -
ID='tapefile-id' PROD INTO=lib.sublib
/*
/&
* $$ EOJ
```

Figure 40. Installing a Product with a Production Part

Installing a Product with a Production and Generation Part

```
// JOB INSTALL PROGRAM PRODUCT    /* INSTALL PROGRAM PRODUCT            */
// ASSGN SYS006,280               /* ASSIGNMENT FOR TAPE UNIT           */
// MTC REW,SYS006                 /* POSITIONING OF THE TAPE            */
// EXEC MSHP,SIZE=1024K
INSTALL PRODUCT FROMTAPE       -  /* CONTROL STATEMENT FOR MSHP         */
  ID = 'prodname...V.R.M'      -  /* SPECIFY IDENTIFIER FOR MSHP        */
  PRODUCTION INTO=prodlib      -  /* SPECIFY PRODUCTION LIBRARY         */
  GENERATION INTO=genlib          /* SPECIFY GENERATION LIBRARY         */
/*
// MTC REW,SYS006                 /* REWIND DISTRIBUTION TAPE           */
/&
```

Figure 41. Installing a Product with a Production and Generation Part

Installing a Product with Selected Parts

The first INSTALL statement installs the required base product; the later step(s) install the selected part(s). Note that the base product is installed before the part requiring the base product. Also, the sequence corresponds to the sequence as backed up to tape.

```
// JOB INSTALL DW/370             /* INSTALL DW/370 BASE AND LEGAL DICT. */
// ASSGN SYS006,280               /* ASSIGNMENT FOR TAPE UNIT            */
// MTC REW,SYS006                 /* POSITIONING OF THE TAPE             */
// EXEC MSHP,SIZE=1024K
INSTALL PRODUCT FROMTAPE       -  /* CONTROL STATEMENT FOR MSHP          */
  ID = 'DW/370BASE.2.1.0'      -  /* SPECIFY TAPE FILE ID OF BASE PROD.  */
  PRODUCTION INTO=PRD2.OFFICE     /* SPECIFY PRODUCTION LIBRARY          */
INSTALL PRODUCT FROMTAPE       -  /* CONTROL STATEMENT FOR MSHP          */
  ID='DW.DI-LEGA.2.1.0'        -  /* SPECIFY TAPEFILEID OF SELECTED DICT.*/
  PRODUCTION INTO=PRD2.OFFICE     /* SPECIFY PRODUCTION LIBRARY          */
/*
// MTC REW,SYS006                 /* REWIND DISTRIBUTION TAPE            */
/&
```

Figure 42. Installing a Product with Selected Parts

Installing a Product from a Stacked Tape

```
// JOB MUESTACK
* INSTALL THE DW/370 2.1.0 BASE AND TWO DICTIONARIES, INSTALL
* CANADIAN FRENCH SUPPORT, FROM A VSE/ESA OPTIONAL PRODUCT TAPE
// PAUSE STACKED TAPE WITH DW/370 2.1.0 BASE AND NLS MOUNTED
// ASSGN SYS006,570
// MTC REW,SYS006
// EXEC MSHP,SIZE=1024K,PARM='PIDSTACKED'
INST PRODUCT FROMTAPE ID='DW/370BASE.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-LEGA.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW.DI-FRA2.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
INST PRODUCT FROMTAPE ID='DW/370FRA2.2.1.0'  -
             PROD INTO=DW210A1.CLRLIB2
/*
/&
```

Figure 43. Installing from a Stacked Tape

Packaging Samples

Chapter 11. Library Member Types

Library members with a type other than OBJ or PHASE are serviced by MSHP as macros only if a one-character type is used. The following list shows the member types allocated for specific use. If product-related data is to be shipped in the product's library, a member type must be chosen that does not conflict with any listed here.

A - Assembler copy books

B - VTAM source books

C - COBOL copy books or C/370 source programs

D - Document Composition Facility image libraries for DCF/DLF VSE NCP Assembler copy books

E - Edited macros (This member type cannot be changed by an APAR fix if you use the new High Level Assembler for VSE. It is recommended not to use this member type, use A-books instead.) Also refer to step 3 on page 80.

F - NCP assembler macros edited

G - ATMS conversion macros for DCF/DLF VSE

H - GML-tags for DCF/DLF VSE or C/370 standard header files

I - ICCF library (DTSFILE) members [3]

M - SPF and other dialog message members

N - SPF and other dialog panel members

P - PL/1 copy books

R - RPG copy books

S - SPF and other dialog skeleton members

T - SPF and other dialog table members

U - Unattended node support members

V - Text repository file (VSE/SP Unique Code only)

W - PC-code for downloading

Y - Information books

Z - Sample programs, installation books

[3] The product-related data must be shipped in the format required for the appropriate sublibrary, that is, data shipped in the I-sublibrary must be of type ICCF DTSFILE member and contain the necessary DTSUTIL control statements (ADD MEMBER, PURGE MEMBER, AND END OF MEMBER control statements).

Library Member Types

Chapter 12. APAR Fix (ZAP)

```
// JOB MSHPZAP
// EXEC MSHP,SIZE=1024K
 CORRECT 5686-010-01-A12:PL54321
 REQUIRES PRE=UD12345
 RESOLVES 'HARDWAIT AT INITIALIZE'
 AFFECTS PHASE=EGQMAIN
 ALTER 468 58E0E004 : 41E00FFF
/*
/&
```

Figure 44. APAR Fix (ZAP) for a Phase

In the following sample a module is corrected, then the linkage editor called to relink the module(s). This results in a corrected phase. PLease note that in addition to the JCL required for a ZAP for a phase, also a work file must be assigned to SYS004.

```
// JOB MSHPZAP
// EXEC MSHP,SIZE=1024K
 CORRECT 5686-010-01-A12:PL54321
 RESOLVES 'HARDWAIT AT INITIALIZE'
 AFFECTS MODULE=EGQMAIN
 ALTER 468 58E0E004 : 41E00FFF
 INVOLVES  LINK=EGQLNK
/*
/&
```

Figure 45. APAR Fix (ZAP) for a Module

For further examples, see *VSE/ESA System Upgrade and Service*, Appendix B.

Service Samples

The next sample ZAPs a macro. Please note that in addition to the JCL required for a simple ZAP for a phase, work files must be assigned to SYS002, SYS003, SYS004.

```
// JOB MSHPZAP
// EXEC MSHP,SIZE=1024K
 CORRECT 5688-143-00-CD1:HB55555
 RESOLVES 'FIX TEXT'
 AFFECTS MACRO=FIJxxxx TYPE=C
 REPLACE : 041910
 A55555     MOVE WS-DOCNO            TO NUMB-FIELD IN SIA2.    041910
/$
 REPLACE : 047010
 A55555     MOVE WS-DOCNO            TO NUMB-FIELD IN SIA2.    047010
/$
/*
/&
```

Figure 46. APAR Fix (ZAP) for a Macro

All samples are REVOKABLE ZAPs, the default parameter on the CORRECT statement. This means the ZAP can be removed using the UNDO statement of MSHP. For more information refer to “Revoking PTFs and APARs” on page 109.

If a ZAP requires additional space, phases and modules can be expanded. However, this expansion remains even if the ZAP is revoked later.

```
// JOB  ZAPPUB2
// EXEC  MSHP
   CORRECT 1234-098-01-VB1 : BI12345 REVOKABLE
   RESOLVES 'MORE THAN 254 I/O DEVICES'
   AFFECTS  PHASE=BMM1 EXPAND=100
*  the EXPAND value is decimal
   ALTER  004080 00000000 : 41E00FFF
   ALTER  004084  . . .   :   . . .
*  assuming 004080 is right at the end of old module or phase
/*
/&
```

Figure 47. APAR Fix (ZAP) Expanding a Phase

Chapter 13. Programming Temporary Fix (PTF)

PTF for Phases

```
// JOB UG00207
*  PROBLEM DESCRIPTIONS:
*    GB00163 - INCORRECT DATA WHEN RUNNING ON
*              SYSTEM EQUIPPED WITH PR/SM FEATURE
*    GB00166 - 3380-K HANDLED AS IBM 3380-E
*    GB00167 - SUPPORT FOR IBM 0671 DASD
*
*  COMPONENT: 5796-PLQ-00-230
*
*  APARS FIXED: GB00163, GB00166, GB00167
*
*  SPECIAL CONDITIONS:
*    COPYRIGHT; (C) COPYRIGHT IBM CORP. 1990
*      LICENSED MATERIAL - PROPERTY OF IBM
*    ENHANCEMENT; VSE/PT SUPPORTS NOW VSE SYSTEMS RUNNING IN AN LPAR
*      ON A MACHINE EQUIPPED WITH THE PR/SM FEATURE
*    ENHANCEMENT; VSE/PT SUPPORTS DASD 0671-MOD4,0671-MOD8
*
*  COMMENTS:
*    NONE;
*
*
// PAUSE EOB OR CANCEL
// EXEC MSHP,SIZE=1024K
 APPLY  5796-PLQ-00-230:UG00207;
  REQUIRES PRE=PLQ230;
  RESOLVES 'PR/SM,3380-K,0671' APARS=(GB00163,GB00166,GB00167);
  AFFECTS                   -
  PHASES =  (  VSEPTSP      -
               VSEPTDA )    -
               VSEPTSA );
  DATA;
 PHASE VSEPTSP,S+X'.......'
 .................PHASE VSEPTSP
 PHASE VSEPTDA,S+X'.......'
 .................PHASE VSEPTDA
 PHASE VSEPTSA,S+X'.......'
 .................PHASE VSEPTSA
/$
/*
/&
```

Figure 48. PTF for Phases

PTF for Modules

```
// JOB UL52789
// OPTION CATAL
*  COMPONENT: 5666-338-01-D75
*  APARS FIXED: PL43691
*  SPECIAL CONDITIONS:
*    COPYRIGHT:            (C) COPYRIGHT IBM CORP.1988
*              LICENSED MATERIAL - PROPERTY OF IBM
*  COMMENTS:
*    CROSS REFERENCE-MODULE/MACRO NAMES TO APARS
*     DDDFO194  PL43691
*
*    CROSS REFERENCE-APARS TO MODULE/MACRO NAMES
*     PL43691   DDDFO194
*
*    THE FOLLOWING MODULES AND/OR MACROS ARE AFFECTED BY THIS PTF:
*
*    MODULES
*     DDDFO194
*
*    LISTEND
// PAUSE EOB OR CANCEL
// EXEC MSHP,SIZE=1024K
APPLY 5666-338-01-D75:UL52789;
REQUIRES PRE=338D75;
REQUIRES PRE=UL47064;
RESOLVES APARS=PL43691;
AFFECTS MODULES=DDDFO194;
INVOLVES LINK=(DDDLEDIT,DDDLSTOR);
DATA;
 ........DDDFO194.OBJ
/$
/*
/&
```

Figure 49. PTF for Modules

PTF for Macros

MSHP recognizes only three types of library members: modules (OBJ), phases (PHASE), and macros (one character extension, refer to Chapter 11, “Library Member Types” on page 131). The following example shows how DisplayWrite/370 CLISTs are serviced as macros. It also shows that the cover letter or comment cards in the PTF may contain instructions that must be read and followed before the fix can be active.

CLISTs of DisplayWrite/370 are shipped as members of a VSE library. When customizing the product, a VSAM file is defined and loaded with the CLISTs from where they are used by DisplayWrite/370. CLISTs can also be updated by a PTF, but this requires a special job to be run after PTF application for updating the VSAM file for the fix to be effective.

```
// JOB UL93576
// OPTION CATAL
*  COMPONENT: 5686-022-34-A43
*  APARS FIXED: PL78703
*  SPECIAL CONDITIONS:
*    COPYRIGHT:            (C) COPYRIGHT IBM CORP.1990
*                LICENSED MATERIAL - PROPERTY OF IBM
*    ACTION:
*      AFTER APPLYING THIS PTF THE FOLLOWING ACTIONS MUST BE TAKEN:
*      1. TRANSFER CLIST DOCUMENT 'DKLARAB' INTO THE DISPLAYWRITE/370
*         VSAM CLUSTER 'DDD210.DDDMAST'.
*         FOR THIS PURPOSE USE THE MODIFIED JOB 'DDDJGLUE.Z' SUPPLIED
*         WITH DW/370 V2.1.0  AND  SPECIFY THE CLIST DOCUMENT NAME IN
*         THE PARAMETER FIELD OF THE EXEC STATEMENT.
*           //   EXEC PGM=DDDDGLUE,SIZE=512K,PARM='DKLARAB'
*      2. IF YOU WORK WITH COMPILED VERSIONS OF THE CLIST DOCUMENTS,
*         REFER TO CHAPTER 5, 'ACTIVATING A COMPILED CLIST DOCUMENT'
*         IN THE DISPLAYWRITE/370 INSTALLATION AND ADMINISTRATION
*         GUIDE, PERFORM STEP 3, 4 AND 5 TO COMPILE AND LOAD THE
*         NEW COPY OF DKLARAB INTO THE DDDCLIX MODULE.
*  COMMENTS:
*    CROSS REFERENCE-MODULE/MACRO NAMES TO APARS
*     DKLARAB   PL78703
*    CROSS REFERENCE-APARS TO MODULE/MACRO NAMES
*     PL78703   DKLARAB
*    THE FOLLOWING MODULES AND/OR MACROS ARE AFFECTED BY THIS PTF:
*    MACROS
*     DKLARAB
*    LISTEND
// PAUSE EOB OR CANCEL
// EXEC MSHP,SIZE=1024K
APPLY 5686-022-34-A43:UL93576 ;
REQUIRES PRE=(022A10,022A43);
SUPERSEDES (UL87283);
RESOLVES APARS=(PL75044,PL78703);
AFFECTS MACROS=(DKLARAB) TYPE=X;
DATA;
 ........DKLARAB.X
/$
/*
/&
```

Figure 50. PTF for Macros

PTF for Synchronizing Service

A PTF for a base product may require a fix in one or more of its features at the same time, so that all products are at the same level. This is achieved by corequisite PTFs.

General description

A PTF for the supervisor of VSE Central Functions requires synchronization. While the compiled supervisors are contained in the VSE Central Functions production system, the supervisor macros (used for compilation) are kept in the VSE Central Functions generation feature, which is a separate product. A PTF replacing a supervisor in the VSE Central Functions production part must, therefore, make sure that the appropriate macros in the generation part are also replaced. The generation feature need, however, not be installed since it is a separate, optional product.

Solution

Using VSE Central Functions 6.1, 5686-066, as an example, this is how PTFs are synchronized:

The supervisor part is component 6 of VSE Central Functions. Thus, the applicable component identifier is 5686-066-06. The CLC for the production system is 15C; the CLC for the generation feature is 15J. The product ID for the production system is 06615C, and for the generation feature 06615J. The fully-qualified component ID for the defective component is 5686-066-06-15C in the production system and 5686-066-06-15J in the generation feature.

Two PTFs are needed: one called UD12345 for component 5686-066-06-15C of the base product; the other PTF UD54321 for the component 5686-066-06-15J of the feature.

The PTF for the feature applies to one environment only and is described by the REQUIRES group with these conditions:

```
PRE= 06615J     the product it fixes must be present
PRE= 06615C     the feature's base product must be present
CO = UD12345    the corresponding PTF for the base must be applied
                together with it.
```

The REQUIRES groups of the PTF for the base product must describe two different environments, one with, one without the feature.

```
PRE= 06615C     the product it fixes must be present
PRE= 06615J     the feature must be present
CO = UD54321    the corresponding PTF for the feature must be
                applied together with it, it is a corequisite PTF.
```

or

```
PRE= 06615C     the product it fixes must be present
NOT= 06615J     the feature must not be present, so no corequisite
                PTF is required.
```

The complete REQUIRES groups for both PTFs:

```
in PTF UD12345                       in PTF UD54321

APPLY 5686-066-06-15C : UD12345      APPLY 5686-066-06-15J : UD54321
REQUIRES PRE = 06615C                REQUIRES PRE = 06615J
REQUIRES PRE = 06615J                REQUIRES PRE = 06615C
REQUIRES CO  = UD54321               REQUIRES CO  = UD12345
OR
REQUIRES PRE = 15C
REQUIRES NOT = 15J
RESOLVES APAR = XX11111              RESOLVES APAR = XX11111
```

Superseding PTF and Associated Requires-Groups

This example shows how the REQUIRES groups for a PTF have to be built if a PTF *supersedes* another PTF.

This scenario deals with a product 5660-012, on release level G40. Its product ID is 012G40. It consists of one component only; the fully qualified component ID is 5660-012-01-G40. It contains the macro BTMOD and several modules M1 to M9.

Now, macro BTMOD needs a fix, to be delivered by PTF numbered UD44444. It is not the first PTF built for the product.

PTF UD11111 replaced modules M1, M7, and M9.
PTF UD22222 replaced modules M6 and M9, and macro BTMOD.
PTF UD33333 replaced macro BTMOD only.
PTF UD44444 replaces again BTMOD only.

Clearly, PTF UD44444 makes PTF UD33333 superfluous, in MSHP terms, it *supersedes* PTF UD33333. But how are the preceding PTFs influencing the last PTF, UD44444?

When PTF UD33333, the superseded one, was shipped it had integrated in BTMOD a fix delivered by its predecessor PTF UD22222. This PTF, in its turn, had not only fixed BTMOD, but also the modules M6 and M9. Module M9 was fixed before by PTF UD11111, together with module M1 and M7. This means, all the fixed modules must be available together with BTMOD, because the code in the new macro BTMOD requires it. This is again ensured by constructing the correct REQUIRES group for UD44444. To do this we look at the REQUIRES groups of its predecessors.

UD11111 REQUIRES PRE=012G40

UD22222 REQUIRES PRE=012G40 REQUIRES PRE=UD11111

UD33333 REQUIRES PRE=012G40 REQUIRES PRE=UD22222

UD44444 REQUIRES PRE=012G40 REQUIRES PRE=UD22222 SUPERSEDES (UD33333)

Of course, it is easy to construct much more complicated REQUIRES groups if more than one component and/or product have to be synchronized, because of a more complex component structure with features on top of the product. This easier example is sufficient to show the rule:

The *superseding* PTF picks up the REQUIRES group(s) of its predecessor in addition to its own - except for pre-reqs already covered. Here: UD11111 is not repeated since it is REQUIREd by UD22222.

REQUIRES-Groups if Several Products Affected

This example shows how the REQUIRES groups for PTFs have to be built if one module is contained in different products. This will keep you from ever implementing such a construction.

The scenario deals with four products, one of them a base product, the other three features on top of this base product. Each of the products contain among other modules MODA, the base product contains in addition also a module called MODB.

An explanation of such a construction could be that module MODA includes more or different functions if included in one of the features, whereas the module with the same name in the base is a stub only. A description of the four products and their relation in MSHP terms is given below.

Base product is identified by the fully qualified component ID=5660-012-01-G40 and product ID=012G40. It contains among others the modules MODA and MODB. Its installation does not require anything special.

Feature 1 is identified by the fully qualified component ID=5660-012-02-G50 and product ID=012G50. It contains among others the module MODA also contained in the base product. The feature requires the base product, but is mutually exclusive with the second feature (012G60). Coded in its history information is REQUIRES PRE=012G40 REQUIRES NOT=012G60

Feature 2 is identified by the fully qualified component ID=5660-012-03-G60 and product ID=012G60. It contains among others the module MODA. The feature requires the base product, but is mutually exclusive with the first feature (012G50). Coded in its history information is REQUIRES PRE=012G40 REQUIRES NOT=012G50

Feature 3 is identified by the fully qualified component ID=5660-012-03-G70 and product ID=012G70. It contains among others the module MODA. The feature requires the base product, is mutually exclusive with the first feature (012G50), and replaces feature 2 (012G60). Coded in its history information is REQUIRES PRE=012G40 REQUIRES NOT=012G50

The replacement of feature 2 is achieved by having the same component ID with different CLC. When installing this feature 3 on top of feature 2, MSHP recognizes this, and issues the message:

G70 supersedes G60. Enter KEEP or DELETE.

If the answer is DELETE, the information for feature 2 is overwritten with the information for feature 3.

Now, this ubiquitous module MODA needs a fix, to be delivered by PTF. In addition, module MODB, contained in the base product only, also needs a PTF.

Since PTFs are built per component, one must check how the four different products can be serviced without applying a PTF to an environment it does not belong to. Below is shown, by help of a truth table, how the PTFs' REQUIRE statements must be coded in order to prevent such a disaster.

The truth table below shows the four products per column and a 1 indicates its presence. Each row shows one environment. Some of these combinations cannot occur because of the different PRE-requisites explained above. This is called an "invalid environment".

Figure 51. Truth Table for Finding the Correct REQUIRES Group

012G40	012G50	012G60	012G70	Required PTF
1	0	0	0	PTF1 REQUIRES PRE=(012G40) REQUIRES NOT=(012G50,012G60,012G70) REQUIRES CO=(PTF5) AFFECTS MODULES=(MODA)
1	0	0	1	PTF2 REQUIRES PRE=(012G70) REQUIRES NOT=(012G50,012G60) REQUIRES CO=(PTF5) AFFECTS MODULES=(MODA)
1	0	1	0	PTF3 REQUIRES PRE=(012G60) REQUIRES NOT=(012G50,012G70) REQUIRES CO=(PTF5) AFFECTS MODULES=(MODA)
1	0	1	1	invalid environment (no PTF)
1	1	0	0	PTF4 REQUIRES PRE=(012G50) REQUIRES NOT=(012G60,012G70) REQUIRES CO=(PTF5) AFFECTS MODULES=(MODA)
1	1	0	1	invalid environment (no PTF)
1	1	1	0	invalid environment (no PTF)
1	1	1	1	invalid environment (no PTF)

Four PTFs are needed to fix MODA, one for each product. The REQUIRES groups are shown in the truth table. A fifth PTF is required to supply the corrected MODB for the base product. Its REQUIRE groups describe the four different environments explained in the truth table:

```
PTF5
  REQUIRES PRE=(012G40)
  REQUIRES NOT=(012G50,012G60,012G70)
  REQUIRES CO=(PTF1)
  OR
  REQUIRES PRE=(012G40,012G70)
  REQUIRES NOT=(012G50,012G60)
  REQUIRES CO=(PTF2)
  OR
  REQUIRES PRE=(012G40,012G60)
  REQUIRES NOT=(012G50,012G70)
  REQUIRES CO=(PTF3)
  OR
  REQUIRES PRE=(012G40,012G50)
  REQUIRES NOT=(012G60,012G70)
  REQUIRES CO=(PTF4)
  AFFECTS MODULES=(MODB)
```

Of course, MODB could have been included in PTF1, which fixes the base component. Nevertheless, a separate PTF5 would be required for the remaining three environments, since the other PTFs are not applied to the base component.

Sample for a Complex PTF Structure

The following samples show two PTFs that have a corequisite relationship. Also shown is the use of the SUPERSEDES statement and the use of REQUIRES groups for two products.

```
// JOB UD90367
// OPTION CATAL
*  COMPONENT: 5686-03-206(032DB6)
*  APARS FIXED: DY43306
*  SPECIAL CONDITIONS:
*    COPYRIGHT:            (C) COPYRIGHT IBM CORP.1993
*               LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
*    PLEASE NOTE THAT THIS PTF HAS COREQUISITES.
*  COMMENTS:
*    CROSS REFERENCE-MODULE/MACRO NAMES TO APARS
*     .....
*    CROSS REFERENCE-APARS TO MODULE/MACRO NAMES
*     ......
*
// PAUSE EOB OR CANCEL
// EXEC MSHP
APPLY 5686-032-06-DB6:UD90367 INDIRECT;
REQUIRES PRE=(032DB6);
REQUIRES NOT=(032DB7);
REQUIRES PRE=(UD49152,UD49206,UD48839);
OR
REQUIRES PRE=(032DB6,032DB7);
REQUIRES PRE=(UD49152,UD49206,UD48839);
REQUIRES CO=(UD90368);
SUPERSEDES (UD48894,UD49112,UD49163,UD49221);
RESOLVES APARS=(DY43103,DY43207,DY43275,DY43295,DY43306);
AFFECTS PHASES=($$A$SUPM,$$A$SUPV,$$A$SUPX,$$A$SUP3,$IJBAR,          -
         $IJBDCMD,$IJBHDUP,......),                                  -
        MACROS=(EXTRACT,GETFLD,IJJLBSER,LBSERV,MAPEXTR,              -
         MODCTB,......) TYPE=E;
```

Figure 52 (Part 1 of 2). PTF for a Base Part of a Component

```
DATA;
 CATALOG   EXTRACT.E  EOD=YY    REPLACE=YES
...macro EXTRACT
 CATALOG   GETFLD.E  EOD=YY    REPLACE=YES
...macro GETFLD
 CATALOG   IJJLBSER.E  EOD=YY    REPLACE=YES
...macro IJJLBSER
 CATALOG   LBSERV.E  EOD=YY    REPLACE=YES
...macro LBSERV
 CATALOG   MAPEXTR.E  EOD=YY    REPLACE=YES
...macro MAPEXTR
 CATALOG   MODCTB.E  EOD=YY    REPLACE=YES
...macro MODCTB
 CATALOG    ......
...macro ......
/$
DATA;
 PHASE $$A$SUPM,+X'000000'
...phase $$A$SUPM
 PHASE $$A$SUPV,+X'000000'
...phase $$A$SUPV
 PHASE $$A$SUPX,+X'000000'
...phase $$A$SUPX
 PHASE $$A$SUP3,+X'000000'
...phase $$A$SUP3
 PHASE $IJBAR,S+X'000000',SVAPFIX
...phase $IJBAR
 PHASE $IJBDCMD,S+X'000000',SVA
...phase $IJBDCMD
 PHASE $IJBHDUP,S+X'000000',SVA
...phase $IJBHDUP
 PHASE ......
...phase ......
/$
/*
/&
```

Figure 52 (Part 2 of 2). PTF for a Base Part of a Component

```
// JOB UD90368
// OPTION CATAL
*  COMPONENT: 5686-03-206(032DB7)
*  APARS FIXED: DY43306
*  SPECIAL CONDITIONS:
*    COPYRIGHT:            (C) COPYRIGHT IBM CORP.1993
*                LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
*    PLEASE NOTE THAT THIS PTF HAS COREQUISITES.
*    ACTION:
*      IN CASE YOU DRIVE YOUR OWN SUPERVISOR YOU HAVE TO RE-ASSEMBLE IT
*      AFTER APPLYING THIS SERVICE.
*      TO GET THE FIXES ACTIVE YOU HAVE TO IPL AFTER SUCCESSFUL
*      ASSEMBLY.
*  COMMENTS:
*    CROSS REFERENCE-MODULE/MACRO NAMES TO APARS
*     .....
*
*    CROSS REFERENCE APARS TO MODULE/MACRO NAMES
*     .....
*
*    THE FOLLOWING MODULES AND/OR MACROS ARE AFFECTED BY THIS PTF:
*
*    MACROS
*     .....
*
*    LISTEND
// PAUSE EOB OR CANCEL
// EXEC MSHP
APPLY 5686-032-06-DB7:UD90368;
REQUIRES PRE=(032DB6,032DB7);
REQUIRES PRE=(UD49222);
REQUIRES CO=(UD90367);
RESOLVES APARS=(DY43306);
AFFECTS MACROS=(DISP,MAPPCE,MAPPUBX,MAPSAACM,MAPTIB,                  -
         ......) TYPE=E;
DATA;
 CATALOG   DISP.E  EOD=YY    REPLACE=YES
...macro DISP
 CATALOG   MAPPCE.E  EOD=YY    REPLACE=YES
...macro MAPPCE
 CATALOG   MAPPUBX.E  EOD=YY    REPLACE=YES
...macro MAPPUBX
 CATALOG   MAPSAACM.E  EOD=YY    REPLACE=YES
...macro MAPSAACM
 CATALOG   MAPTIB.E  EOD=YY    REPLACE=YES
...macro MAPTIB
 CATALOG   ......
...macro ......
/$
/*
/&
```

Figure 53. PTF for a Generation Part of a Component

Chapter 14. Shipping PC Code with VSE

The following information applies to VSE 1.3 and later. If you use a previous version of VSE, please refer to “Using VSE/ESA before 1.3” on page 146.

Shipping Workstation Code with VSE/ESA

The introduction of workstation file transfer to/from the VSE libraries in VSE/ESA 1.3.0 significantly simplifies the procedure to distribute workstation products with the VSE/ESA system. Theoretically, the product owner could simply send the files from the workstation to a VSE/ESA library as a binary string of data, and the user would download the files from the VSE library to his workstation; this could be done with SEND or RECEIVE as shown in the following example, or with the equivalent function of the emulator that is used.

```
SEND    c\product.EXE c: product EXEBIN (FILE=LIB BINARY L=lib S=sublib
RECEIVE c\product.EXE c: product EXEBIN (FILE=LIB BINARY L=lib S=sublib
```

So far the good news. Now here is some bad news: If the workstation product is to be **serviced** on VSE/ESA using **MSHP PTFs**, then the following restrictions apply:

- The workstation files must be stored in the VSE library in fixed-80 logical record format.
- The members in the VSE library can only have a one-character member type.

Fixed-80 record format can easily be achieved with the LRECL option of the SEND command.

The one-character member type is an MSHP restriction. (Chapter 11, “Library Member Types” on page 131 lists the member types allocated for specific use.) Unfortunately it forbids a one-to-one mapping of the member names unless the workstation files have unique file names independent of the file extension. It is the product owner's responsibility to devise a suitable naming scheme for the workstation and VSE library members.

Packaging Workstation Code into a Product Library

The following sample procedure assumes that:

- The workstation product is to be serviced with MSHP and fixed-80 record format is required.
- The product code exists on a workstation attached to a VSE/ESA 1.3.0 host or higher.
- The product is installed in sublibrary **PWS.PROD**.
- Product members are stored with member type **X** (refer to Chapter 11, “Library Member Types” on page 131 for reserved member types).

Proceed as follows:

1. Switch to the host session and sign on to the VSE/ESA system.
2. Prepare the host session for file transfer (PF6 or fast path 386 from Main Selection Panel).
3. Send product members to the VSE library, for example:

```
SEND  file1.CMD  c: file1  X  (FILE=LIB  L=PWS  S=PROD  BINARY  LRECL=80
SEND  file2.EXE  c: file2  X  (FILE=LIB  L=PWS  S=PROD  BINARY  LRECL=80
SEND  file3.ZIP  c: file3  X  (FILE=LIB  L=PWS  S=PROD  BINARY  LRECL=80
```

The result of the these commands is three files FILE1.X, FILE2.X, and FILE3.X in VSE library PWS.PROD.

Note that this format of the SEND commands applies to all kinds of workstation files, that is, ASCII text files, binary files, compressed files, etc.

Also note that, if the size of the PWS file is not a multiple of 80, the last record is padded with ASCII blanks X'20'. When such a file is received back to the workstation, it will contain these extra blanks. These blanks can be removed by a separate program if they cause problems. Please let us know if you find any problems with these extra blank characters at the end of a file.

4. Prepare the product tape as described in Chapter 7, “Creating Installation Tapes” on page 79.

Note: It is recommended to send ASCII text in binary form to avoid code page conversion problems.

The Download Procedure

1. Switch to the host session and sign on to the VSE/ESA system.
2. Prepare the host session for file transfer (PF6 or fast path 386 from Main Selection Panel).
3. Download the product members with the corresponding RECEIVE commands, for example:

```
RECEIVE  file1.CMD  c: file1  X  (FILE=LIB  L=PWS  S=PROD  BINARY
RECEIVE  file2.EXE  c: file2  X  (FILE=LIB  L=PWS  S=PROD  BINARY
RECEIVE  file3.ZIP  c: file3  X  (FILE=LIB  L=PWS  S=PROD  BINARY
```

The RECEIVE commands are identical with the SEND commands except for the LRECL option that does not apply.

This facility is part of the VSE Workstation File Transfer Support --formerly called Intelligent Workstation Support (IWS)-- which is a CICS application and is included in VSE from VSE/SP 3.2.0 on.

If you need this facility already on VSE/SP Version 2 or additional functions from what is already included in VSE, you can write your own routines and ship them with your product.

Note: VSE/SP 2.1.4 is the earliest VSE release to support such downloadable routines.

Automatic Download

The steps described previously can easily be put into a procedure, and automated:

1. Load download procedure (which loads the other files)
2. Run download procedure

Using VSE/ESA before 1.3

The procedure to distribute workstation products with VSE/ESA before 1.3 has a few restrictions:

- You need to preprocess the members before they are stored in a VSE library.
- You need to define a DFHPPT entry for each member.
- There is a size limit for the members due to CICS storage restraints.
- You need to define the product library in the LIBDEF chain of the CICS partition.

Packaging PC Code into a Product Library

If you want to distribute your PC code along with your host application, do the following:

1. Have your PC code on your VSE system in executable PC ASCII format, for example, by sending the files from a PC to the host in binary format using the VM or TSO file transfer function (IND$FILE).
2. Convert each PC member into an Assembler source program consisting of hexadecimal DC constants. A sample REXX EXEC is shown in “Sample REXX EXEC to Convert PC Code” on page 153.
3. Assemble the source programs (for example, with the CMS Assemble function).
4. Link edit and catalog each program as a separate phase in the corresponding VSE product library. Use phase names that are in line with the naming conventions for this product. In other words, these 'PC phases' are to be treated like any other phases of the product.
5. Prepare download instructions for the user, that is, the CICS PPT entries to be made for each phase and the RECEIVE commands to be issued, including message and return codes. A sample set of download instructions is given in the following section.

Downloading PC Code from the VSE Host to the PC

The following is a sample set of instructions that demonstrates how the user can download PC members shipped with the VSE system. The sample assumes:

- The product is installed in VSE library PRD2.PROD.
- There are two phases XXXPC1 and XXXPC2, which should be downloaded to the PC as SAMPLE.EXE and SAMPLE.PIC.

Note that the sample uses the VSE program CFTRDOWN, which is described in “Logic of CFTRDOWN” on page 150. As mentioned under “Shipping VM Code with a VSE Product” on page 89, starting with VSE/SP 3.2, VSE contains the user exit program CFTRDOWN, which offers a general download facility. CFTRDOWN uses the official user exit interface for file transfer from CICS temporary storage, as described in the *VSE/ESA Administration* manual. Therefore, if download requirements are not covered by CFTRDOWN or if a download routine is already needed on an earlier release than VSE/SP 3.2, you may write and ship your own download programs.

Preparing for Download:

1. After successful installation of the product, you should make sure that library PRD2.PROD is included in the LIBDEF phase chain active for the CICS partition (for example, 0 LIBLIST PHASE,F2).
2. Include two entries in the CICS PPT:

   ```
   DFHPPT TYPE=ENTRY,RSL=PUBLIC,PROGRAM=XXXPC1
   DFHPPT TYPE=ENTRY,RSL=PUBLIC,PROGRAM=XXXPC2
   ```

   You can either update and submit member DFHPPTSP in ICCF library 59 or use the CICS RDO facility.
3. If you reassembled DFHPPTSP, you must shutdown and restart CICS.
4. The download operation uses CICS temporary storage queue CFTRnnnn, where nnnn is your CICS terminal ID. If this queue already contains data, save this data before download, if necessary.
5. Sufficient CICS temporary storage on disk (DFHTEMP) is required to accommodate the largest member to be downloaded. Supply this amount of space in your documentation.
6. At your PC, make sure you have sufficient disk or diskette space to download the PC members. Supply these values in your product documentation.

The Download Procedure:

1. Switch to the host session and sign on to the VSE system.
2. Press PF6 to escape to native CICS or, if you are not authorized to do so, select 387 for file transfer mode.
3. Switch back to the PC session and enter the command:

   ```
   RECEIVE SAMPLE.EXE XXXPC1 (FILE=TS BINARY PROGRAM=CFTRDOWN
                          or
   RECEIVE SAMPLE.EXE XXXPC1 [ (FILE=TS PROGRAM=CFTRDOWN  for 5550PCs
   ```

   After you receive message

   ```
   INW0025I File received from TS queue CFTRnnnn
   ```

   enter the second command

   ```
   RECEIVE SAMPLE.PIC XXXPC2 (FILE=TS BINARY PROGRAM=CFTRDOWN
                          or
   RECEIVE SAMPLE.PIC XXXPC2 [ (FILE=TS PROGRAM=CFTRDOWN  for 5550PCs
   ```

   When you receive message INW0025I again, the download process has been successfully completed.

Download Errors: During download, the following errors may occur:

INW0032I Failure to link to program CFTRDOWN

Explanation: Program CFTRDOWN was not found in any of the VSE libraries specified in the library chain active for the CICS partition, or there is no PPT entry, or the PPT entry is disabled for this program, or the PPT entry does not specify RSL=PUBLIC.

INW0027I File transfer terminated by program CFTRDOWN. RC=nnnn

Explanation: An error was detected by the download program CFTRDOWN indicated by the return code **nnnn**:

1000 The member specified as host file name in the RECEIVE command was not found in any of the VSE libraries specified in the library chain active for the CICS partition, or there is no PPT entry, or the PPT entry is disabled for this member.

2000 Security violation. The member to be downloaded was not defined with RSL=PUBLIC in the PPT.

3000 CICS temporary storage space on disk is exhausted. Download uses temporary storage as intermediate storage while there is not enough space to accommodate the entire member to be downloaded.

4000 Security violation. The user is not authorized to access CICS temporary storage queues named CFTRxxxx.

5000 I/O error. An irrecoverable I/O error occurred during writing to CICS temporary storage on disk.

6000 Invalid segment size. The segment size specified in the RECEIVE command is not a number between 80 and 32600

00xx An unexpected error occurred during writing to CICS temporary storage. xx is the EIBRESP code returned by CICS for the WRITEQ TS command.

INW0036I Security violation. Failure to access a protected resource

Explanation: This message occurs if a program like CFTRDOWN or the phase to be downloaded is not defined with RSL=PUBLIC in the CICS PPT, or the user is not authorized to access CICS temporary storage queues CFTRxxxx.

Downloading Large Amounts of PC Data: If a large number of PC members is to be downloaded, the product owner should provide a procedure containing all RECEIVE commands rather than requiring the user to type in these commands.

For example, the product owner:

1. Prepares at the PC a procedure DOWNLOAD.BAT, which contains all necessary RECEIVE commands:

   ```
   RECEIVE PC1 HOST1 (FILE=TS BINARY PROGRAM=CFTRDOWN
   RECEIVE PC2 HOST2 (FILE=TS BINARY PROGRAM=CFTRDOWN
       .
       .
   RECEIVE PCn HOSTn (FILE=TS BINARY PROGRAM=CFTRDOWN
   ```

   Note that the last RECEIVE command in the procedure should download the smallest member in order to avoid large amounts of data occupying CICS temporary storage after download.

2. Sends this procedure to his VM or TSO host in binary format.
3. At the host, converts and packages the procedure into a VSE library, as described in the preceding section.

The user then types the following command to download the procedure and then execute DOWNLOAD to receive all files:

```
RECEIVE DOWNLOAD.BAT HOSTNAME (FILE=TS BINARY PROGRAM=CFTRDOWN
```

Download Considerations for IBM 5550PC Users: File transfer on 5550Pcs (widely used in the DBCS countries) is different from normal PCs in the following areas:

1. The BINARY option is not supported; BINARY/NOCRLF is the default.
2. The left parenthesis preceding the options in the SEND/RECEIVE commands must be preceded by an additional left bracket [.

Therefore, separate procedures must be supplied for 5550PC users, including the left bracket, and omitting the BINARY option.

Logic of CFTRDOWN

```
/*  START OF SPECIFICATIONS ****************************************/
/*                                                                  */
/*01* MODULE-NAME = CFTRDOWN  (CMS FILENAME: INWPDOWN)              */
/*                                                                  */
/*01*  DOWNLOAD PC CODE FROM VSE LIBRARY                           */
/*                                                                  */
/*01* STATUS =  NEW IN VSE/SP 3.2                                  */
/*                                                                  */
/*01* FUNCTION = THIS PROGRAM IS CALLED AS A USER EXIT DURING A    */
/*               DOWNLOAD (RECEIVE) OPERATION FROM THE HOST TO THE  */
/*               PC IN ORDER TO DOWNLOAD PC MEMBERS  STORED AS      */
/*               PHASES IN A VSE/SP SUBLIBRARY TO A PC DISKETTE OR  */
/*               OR HARD DISK. FOR EXAMPLE THE COMMAND              */
/*                                                                  */
/*      RECEIVE PCFILE PHASE1 (FILE=TS BINARY PROGRAM=CFTRDOWN      */
/*                                                                  */
/*               WILL CAUSE THE FILE TRANSFER PROGRAM TO LINK TO    */
/*               'CFTRDOWN' WHICH LOADS 'PHASE1' FROM THE VSE/SP    */
/*               LIBRARY INTO MAIN STORAGE AND WRITES ITS CONTENTS  */
/*               IN 80-BYTE SEGMENTS INTO A CICS TS QUEUE WHICH IS  */
/*               THEN DOWNLOADED INTO 'PCFILE' AT THE PC.           */
/*                                                                  */
/*               IF A SEGMENT SIZE OTHER THAN 80 BYTES IS REQUIRED, */
/*               THIS CAN BE SPECIFIED AS A COMMENT IN THE RECEIVE  */
/*               COMMAND, FOR EXAMPLE:                              */
/*                                                                  */
/*    RECEIVE PCFILE PHASE1 (FILE=TS BINARY PROGRAM=CFTRDOWN) 1024  */
/*                                                                  */
/*               WILL CAUSE CFTRDOWN TO BREAK PHASE 'PHASE1'        */
/*               INTO 1024-BYTE SEGMENTS. ANY NUMBER BETWEEN 80     */
/*               AND 32600 MAY BE SPECIFIED FOLLOWING THE RIGHT     */
/*               PARENTHESIS, WITH OR WITHOUT LEADING BLANKS.       */
/*                                                                  */
/*01* NOTES = NONE                                                 */
/*                                                                  */
/*02*   DEPENDENCIES = 1. THE PC MEMBERS MUST HAVE BEEN CONVERTED   */
/*                        TO VSE PHASE FORMAT. THIS CAN, FOR EXAMPLE*/
/*                        BE DONE BY A REXX EXEC WHICH CONVERTS THE */
/*                        PC CODE INTO AN ASSEMBLER SOURCE PROGRAM  */
/*                        CONSISTING OF DC CONSTANTS.               */
/*                     2. THE VSE LIBRARY THAT CONTAINS THE PC      */
/*                        MEMBERS TO BE DOWNLOADED MUST BE SPECIFIED*/
/*                        IN THE ACTIVE PHASE LIBRARY CHAIN OF THE  */
/*                        CICS PARTITION.                           */
/*                     3. THE NAMES OF THE PHASES TO BE DOWNLOADED  */
/*                        MUST BE INCLUDED IN THE ACTIVE CICS PPT   */
/*                        WITH RSL=PUBLIC.                          */
/*                                                                  */
/*                                                                  */
/*02*   ATTRIBUTES = REENTRANT                                     */
/*                                                                  */
/*01* ENTRY-POINT = INWPDOWN                                       */
/*                                                                  */
```

```
/*01* CALLED BY   = MODULE INWPGTS OF THE FILE TRANSFER PROGAM      */
/*                  INWPCCOM WHEN PROGRAM=CFTRDOWN WAS SPECIFIED    */
/*                  IN A RECEIVE COMMAND TO TEMPORARY STORAGE.      */
/*                                                                  */
/*01* INPUT = USER EXIT COMMUNICATION AREA:                         */
/*            *        CHAR(2)   RESERVED                          */
/*            URETCODE FIX(15)   ERROR RETURN CODE (X'00' ON INPUT) */
/*            UFILE    CHAR(8)   NAME OF PHASE TO BE DOWNLOADED     */
/*            UQUEUE   CHAR(8)   NAME OF TS QUEUE TO BE USED        */
/*            UOPTION  CHAR(8)   OPTIONALLY, THE SEGMENT SIZE SPE-  */
/*                               CIFIED IN THE RECEIVE COMMAND.     */
/*                               BETWEEN 80 AND 32600.              */
/*                                                                  */
/*01* OUTPUT = CONTENTS OF PHASE 'UFILE' STORED IN CICS TS QUEUE    */
/*             'UQUEUE' IN N-BYTE SEGMENTS (ITEMS), WHERE N IS 80   */
/*             (DEFAULT) OR THE NUMBER SPECIFIED IN UOPTION         */
/*                                                                  */
/*01* ERRORS = IN CASE OF ERRORS A RETURN CODE IS PASSED BACK TO    */
/*             TO CALLING MODULE INWPGTS IN FIELD 'URETCODE':       */
/*             1000 -  PGMIDERR                                     */
/*                     PHASE NOT FOUND BECAUSE IT IS                */
/*                     NOT IN LIBRARY, OR                           */
/*                     LIBRARY NOT IN ACTIVE LIBRARY CHAIN, OR      */
/*                     NOT DEFINED IN PPT, OR                       */
/*                     PPT ENTRY IS DISABLED.                       */
/*             2000 -  NOTAUTH                                      */
/*                     PHASE NOT DEFINED WITH RSL=PUBLIC            */
/*             3000 -  NOSPACE                                      */
/*                     CICS TEMPORARY STORAGE ON DISK EXHAUSTED     */
/*             4000 -  NOTAUTH                                      */
/*                     RESOURCE SECURITY CHECKING IS ACTIVE FOR     */
/*                     ACTIVE FOR CICS TS QUEUES CFTRXXXX, BUT THE  */
/*                     REQUESTING USER ID DOES NOT HAVE THE         */
/*                     REQUIRED SECURITY KEY.                       */
/*             5000 -  IOERR                                        */
/*                     AN IRRECOVERABLE I/O ERROR OCCURRED DURING   */
/*                     WRITEQ TS                                    */
/*             6000 -  INVALID SEGMENT SIZE                         */
/*                     THE SIZE SPECIFIED IN UOPTION MUST BE AN     */
/*                     EBCDIC NUMBER BETWEEN 80 AND 32600.          */
/*             00XX -  AN UNEXPECTED ERROR OCCURRED DURING WRITEQ   */
/*                     OR DELETEQ TS; XX IS THE EIBRESP CODE        */
/*                     RETURNED BY CICS.                            */
/*             THE RETURN CODE IS DISPLAYED IN FILE TRANSFER MSG:   */
/*             INW0027I FILE TRANSFER TERMINATED BY PROGRAM         */
/*                      CFTRDOWN. RC=XXXX                           */
/*                                                                  */
/*01* EXIT-NORMAL = RETURN TO CALLER WITH URETCODE=0000             */
/*                                                                  */
/*01* EXIT-ERROR =  RETURN TO CALLER WITH URETCODE=XXXX (SEE ABOVE) */
/*                                                                  */
/***  END OF SPECIFICATIONS  ***************************************/
```

```
/*********************************************************************/
/*                    L O G I C                                      */
/* CHECK SEGMENT SIZE IN UOPTION                                     */
/* IF SEGMENT SIZE NOT BETWEEN 80-32600 THEN ERROR CODE = 6000       */
/* IF ERROR CODE = 0 THEN DO_0                                       */
/* LOAD THE INPUT PHASE INTO MAIN STORAGE                            */
/* IF LOAD WAS SUCCESSFUL THEN DO_1                                  */
/*    DELETE EXISTING TS QUEUE                                       */
/*    IF NO ERROR OCCURRED THEN DO_2                                 */
/*       EXEC CICS SYNCPOINT                                         */
/*       WRITE HEADER RECORD TO TS QUEUE                             */
/*       IF WRITE HEADER WAS SUCCESSFUL THEN DO_3                    */
/*          DO UNTIL ENTIRE PHASE PROCESSED                          */
/*             WRITE 80-BYTE PORTION OF PHASE TO TS QUEUE            */
/*             IF WRITE FAILED THEN                                  */
/*                LEAVE DO_3                                         */
/*          END                                                      */
/*          IF THERE IS A REMAINDER                                  */
/*             WRITE REMAINDER TO TS QUEUE                           */
/*       END DO_3                                                    */
/*       IF WRITE TO TS QUEUE FAILED THEN                            */
/*          RETURN WITH ERROR CODE 3000 4000 5000 OR 00XX            */
/*    END DO_2                                                       */
/*    ELSE                                                           */
/*    RETURN WITH ERROR CODE 4000 OR 00XX                            */
/* END DO_1                                                          */
/* ELSE                                                              */
/* RETURN WITH ERROR CODE 1000                                       */
/* END DO_0                                                          */
/* ELSE                                                              */
/* RETURN WITH ERROR CODE 6000                                       */
/*********************************************************************/
```

Sample REXX EXEC to Convert PC Code

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                     P C P A C K                                   */
/* This exec converts executable PC code into an Assembler source    */
/* program consisting of hex DC constants.                           */
/* The assembled program can then be cataloged in a VSE/SP phase     */
/* sublibrary for subsequent downloading to a PC.                    */
/*                                                                   */
/* Invocation:  PCPACK filename filetype (filemode)                  */
/* Input:       CMS file containing the executable PC member         */
/*              in fixed or variable record format                   */
/* Output:      CMS file ' filename ASSEMBLE filemode' containing    */
/*              the converted PC member in the form of an            */
/*              Assembler source program                             */
/*                                                                   */
/* Downloading: The name of the cataloged phase must be included in  */
/*              CICS PPT of the VSE/SP system with RSL=PUBLIC.       */
/*              The phase can then be downloaded to the PC with      */
/*                                                                   */
/*         RECEIVE pcname phasename (FILE=TS BINARY PROGRAM=CFTRDOWN */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
                                                                  /*   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* check input                                                       */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
arg fn ft fm
if fn = "" | ft = "" then do
   say " COMMAND FORMAT IS: PCPACK filename filetype (filemode) "
   exit
end
if fm = "" then fm = a
'ERASE' fn 'ASSEMBLE' fm
                                                                  /*   */
```

Figure 54 (Part 1 of 3). REXX EXEC Sample to Convert PC Code

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* write TITLE and START statement                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
outline = ,
"         TITLE   '"fn" - PC PROGRAM'"
push outline
'EXECIO 1 DISKW' fn 'ASSEMBLE' fm '0 F'
outline = ,
fn"  START"
push outline
'EXECIO 1 DISKW' fn 'ASSEMBLE' fm
                                                                  /*   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* read first input line                                             */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
restlen = 0
reststring = ""
'EXECIO 1 DISKR' fn ft fm '(VAR' inline
readrc = RC
                                                                  /*   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* loop to read all lines                                            */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
DO WHILE readrc ¬= 2
   reclen = restlen + LENGTH(inline)
   totline = reststring||inline
                                                                  /*   */
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  */
   /* Loop to process 16-byte portions of each input line.           */
   /* The length 16 is chosen because this allows - in case of       */
   /* problems - an easy comparison of the output of this exec       */
   /* with the original PC member using the PC DOS debug function.   */
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  */
   DO i = 1 by 16 WHILE reclen ►= 16
      outstring = SUBSTR(totline,i,16)
      hexstring = C2X(outstring)
      outline   = ,
"         DC    X'"||hexstring||"'"
      push outline
      'EXECIO 1 DISKW' fn 'ASSEMBLE' fm
      reclen = reclen - 16
   END
                                                                  /*   */
```

Figure 54 (Part 2 of 3). REXX EXEC Sample to Convert PC Code

```
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
   /* get remainder of input line                                       */
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
   restlen = reclen
   IF restlen ► 0 then
      reststring = SUBSTR(totline,i,restlen)
   ELSE
      reststring = ""
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
   /* read next input line                                              */
   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
   inline = "INLINE"
   'EXECIO 1 DISKR' fn ft fm '(VAR' inline
   readrc = RC
END
                                                                    /*   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* output remainder of last input line                               */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
IF restlen ► 0 then DO
      hexstring = C2X(reststring)
      outline   = ,
"         DC    X'"||hexstring||"'"
      push outline
      'EXECIO 1 DISKW' fn 'ASSEMBLE' fm
END
                                                                    /*   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* output last line :  END fn                                        */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
outline = ,
"         END "Fn
push outline
'EXECIO 1 DISKW' fn 'ASSEMBLE' fm  '(FINIS'
SAY "FILE " FN "ASSEMBLE" FM " CREATED"
```

Figure 54 (Part 3 of 3). REXX EXEC Sample to Convert PC Code

Downloading PC Code

Chapter 15. Job for Customizing

For a discussion of this sample, please refer to “Writing Customizing Jobs” on page 97.

```
..* $$ JOB JNM=IPZINST,DISP=D,CLASS=0
..* $$ PRT DISP=D,CLASS=A
..* $$ PUN DISP=I,CLASS=A
// JOB IPZINST
* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
* ** THE JOB IPZINST                                                  **
* **  o  INSTALLS THE VSAM CLUSTERS OF DM/VSE                         **
* **                                                                  **
* **  o  SUBMITS THE FOLLOWING JOBS:                                  **
* **     IPZCAFCT - CATALOGS FCT ENTRIES FOR                          **
* **                SELECTED DICTIONARIES                             **
* **     IPZCMFCT - CATALOGS FCT ENTRIES FOR MIGRATION                **
* **                                                                  **
* **  o  ASSEMBLES AND LINKS CICS BMS MAPS AND                        **
* **     THE DM/VSE DISOSS MIGRATION PROGRAM AND                      **
* **     LOADS DM/VSE SAMPLE PANELS.                                  **
* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
* ** IF YOU DON'T INSTALL DM/VSE THE FIRST TIME                       **
* ** MAKE SURE THAT THE FOLLOWING FILES ARE CLOSED:                   **
* **     DDD*                                                         **
/. C     DDDMAST  DDDPDOC  DDDWDOC  DDDRCO   DDDSLG                   **
/. C     DDDTRA   DDD2DOC  DDDL3GX                                    **
* **     EKL*                                                         **
/. C     EKLAFRI  EKLBPOR  EKLDANS  EKLDEUT  EKLDSCH                  **
* **     DKL*                                                         **
/. C     DKLFONT                                                      **
* **     IPZ*                                                         **
/. C     IPZDFLT  IPZREST                                             **
* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// PAUSE PRESS ENTER TO CONTINUE
```

Figure 55 (Part 1 of 5). Job for Customizing

```
/. C  ************************************************************
/. C  *  THE FOLLOWING MODIFICATIONS M U S T BE DONE BEFORE RUNNING *
/. C  *  THE JOB IPZINST:                                          *
/. C  *  --  IN EACH LINE DELETE THE STRING '..' WHEN IT APPEARS IN *
/. C  *      THE FIRST TWO COLUMNS.                                *
/. C  *      (SEARCH FOR ..* AND ../)                              *
/. C  *  --  REPLACE THE STRING '-V001-' BY ONE OR MORE VSAM VOLUME *
/. C  *      IDS OWNED BY THE VSAM CATALOG THAT CONTAINS THE       *
/. C  *      VSAM FILES OF DM/VSE.                                 *
/. C  *  --  MODIFY THE FOLLOWING PARAMETERS IF NECESSARY:         *
/. C  *                                                            *
/. C  *      ---------------------------------------------------   *
/. C  *      |PARAMETER | PARAMETER| DEFAULT  | DESCRIPTION      |  *
/. C  *      |NAME      | TYPE     | VALUE    |                  |  *
/. C  *      |----------|----------|----------|------------------|  *
/. C  *      |LIB       | SETPARM  | PRD2     | LIBRARY NAME     |  *
/. C  *      |----------|----------|----------|------------------|  *
/. C  *      |SUBLIB    | SETPARM  | PROD     | SUBLIBRARY NAME  |  *
/. C  *      |----------|----------|----------|------------------|  *
/. C  *      |TAPE      | SETPARM  | 180      | TAPE UNIT ADDRESS|  *
/. C  *      |----------|----------|----------|------------------|  *
/. C  *      |CATNAME   | SETPARM  | VSESPUC  | CATALOG NAME     |  *
/. C  *      ---------------------------------------------------   *
/. C  *                                                            *
/. C  *  --  MODIFY THE FOLLOWING VALUES IF NECESSARY:             *
/. C  *                                                            *
/. C  *      ---------------------------------------------------   *
/. C  *      |USED VALUE           |WHERE VALUE IS USED         |  *
/. C  *      |---------------------|----------------------------|  *
/. C  *      |PRD2.PROD            | LIBR COMMANDS,             |  *
/. C  *      |                     | LIBDEF STATEMENTS          |  *
/. C  *      |---------------------|----------------------------|  *
/. C  *      |VSESP.USER.CATALOG   | VSAM CATALOG ID            |  *
/. C  *      ---------------------------------------------------   *
/. C  *                                                            *
/. C  *  --  MODIFY THE SETPARM PARAMETERS LISTED IN THE PARAMETER  *
/. C  *      LIST (USER SELECTION) TO YOUR NEEDS.                  *
/. C  ************************************************************
/. C  * NOTE:                                                     *
/. C  * -->  "PRD2.PROD" IS ASSUMED TO BE THE LIBRARY WHERE YOU    *
/. C  *      RESTORE THE CONTENTS OF THE TAPE DM/VSE120 BASE1OF3   *
/. C  *      AND DM/VSE120 BASE3OF3.                               *
/. C  *      IF YOU USE A DIFFERENT SUBLIBRARY MODIFY              *
/. C  *        o  THE LIB AND SUBLIB PARAMETER (USER SELECTION)    *
/. C  *      LOCATE "PRD2.PROD" AND MODIFY                         *
/. C  *        o  ONE LIBR STATEMENT TO CATALOG IPZFCTM6/7 (PART1.6)*
/. C  *        o  ONE LIBR STATEMENT TO CATALOG DDDLEX16/17 (PART 6)*
/. C  * -->  ALL DM/VSE FILES WILL BE CREATED IN VSAM MANAGED SPACE.*
/. C  *      IF YOU USE DIFFERENT VSAM CATALOGS MODIFY THE JOB     *
/. C  *      IPZINST TO YOUR NEEDS.  (FOR EXAMPLE, DLBL, SETPARM   *
/. C  *      (STATEMENTS AND THE VSAM CATALOG ID                   *
/. C  *       IN VSAM STATEMENTS).                                 *
/. C  ************************************************************
```

Figure 55 (Part 2 of 5). Job for Customizing

```
/. C
/. C  ************************************************************
/. C  ************************************************************
/. C  *                                                          *
/. C  *  START OF PARAMETER LIST   (USER SELECTION)              *
/. C  *                                                          *
/. C  ************************************************************
/. C  ************************************************************
/. C
/. C  * ---------------------------------------------------------*
/. C  * CHANGE THE PARAMETERS TO YOUR NEEDS.                     *
/. C  * ---------------------------------------------------------*
/. C  LIB.SUBLIB CONTAINS DM/VSE (DM/VSE120 BASE1OF3 AND BASE3OF3)
/. C  THE LIB AND SUBLIB PARAMETER ARE ALSO USED FOR THE
/. C  LIBR COMMANDS IN PART3 (LANGUAGE SELECTION).
// SETPARM LIB=PRD2              LIBRARY
// SETPARM SUBLIB=PROD           SUBLIBRARY
/. C
/. C  FOLLOWING LIBDEF STATEMENTS ARE ACTIVE FOR THE ENTIRE JOB
// LIBDEF *,SEARCH=&LIB..&SUBLIB
// LIBDEF PHASE,CATALOG=&LIB..&SUBLIB
/. C
// SETPARM TAPE=180               TAPE UNIT ADDRESS (USED FOR THE TAPE
/. C                              DM/VSE120 BASE2OF3)
// SETPARM CATNAME=VSESPUC        CATALOG NAME FOR VSAM CLUSTER
/. C
/. C  * ---------------------------------------------------------*
/. C  * SELECT THE DM/VSE INSTALLATION LEVEL:                    *
/. C  * TYPE 0   IF THIS IS THE FIRST INSTALLATION OF DM/VSE     *
/. C  * TYPE 110 IF YOU HAVE ALREADY INSTALLED DM/VSE 1.1.0     *
/. C  * TYPE 111 IF YOU HAVE ALREADY INSTALLED DM/VSE 1.1.1     *
/. C  * TYPE 112 IF YOU HAVE ALREADY INSTALLED DM/VSE 1.1.2     *
/. C  * TYPE 120 IF YOU HAVE ALREADY INSTALLED DM/VSE 1.2.0     *
/. C  * ---------------------------------------------------------*
/. C
// SETPARM ILEVEL=0               DM/VSE INSTALLATION LEVEL
/. C
/. C  * ---------------------------------------------------------*
/. C  * SELECT THE CICS/DOS/VS INSTALLATION LEVEL:               *
/. C  * TYPE 6   IF YOU INSTALL DM/VSE ON A VSE/SP WITH CICS 1.6 *
/. C  * TYPE 7   IF YOU INSTALL DM/VSE ON A VSE/SP WITH CICS 1.7 *
/. C  * ---------------------------------------------------------*
/. C
// SETPARM JLEVEL=7              CICS/DOS/VS INSTALLATION LEVEL
/. C
/. C  * ---------------------------------------------------------*
```

Figure 55 (Part 3 of 5). Job for Customizing

```
/. C  * SELECT THE LANGUAGE BY TYPING YES/NO.                        *
/. C  * US ENGLISH IS REQUIRED AND ALWAYS INSTALLED.                 *
/. C  * EACH LANGUAGE SELECTION CAUSES AUTOMATICALLY A DICTIONARY   *
/. C  * SELECTION.                                                  *
/. C  * NOTE: YOU CAN LOAD A LANGUAGE ONLY IF THE SETPARM PARAMETER *
/. C  *       "PART3" IS SET TO YES.                                 *
/. C  *       TAKE CARE WHEN RELOADING A LANGUAGE,                   *
/. C  *       FOR EXAMPLE, MODIFICATIONS OF CLISTS AREL LOST.        *
/. C  * -------------------------------------------------------------*
/. C
// SETPARM BPORT=NO                BRAZIL       LANGUAGE SELECTION
// SETPARM DANSK=NO                DANISH       LANGUAGE SELECTION
// SETPARM DEUTS=NO                GERMAN       LANGUAGE SELECTION
/. C  * -------------------------------------------------------------*
/. C  * SELECT DICTIONARIES BY TYPING YES/NO                         *
/. C  * (US ENGLISH IS ALWAYS INSTALLED)                            *
/. C  * NOTE: YOU CAN LOAD A DICTIONARY ONLY IF SETPARM PARAMETER   *
/. C  *       "PART4" IS SET TO YES.                                 *
/. C  *       A SELECTION OF DICTIONARIES CAUSES THE JOB IPZCAFCT    *
/. C  *       TO CATALOG IPZFCT2.A OR IPZFCT3.A (FCT ENTRIES).       *
/. C  *       IF A DICTIONARY IS SELECTED THE CORRESPONDING          *
/. C  *       FCT ENTRY IS ADDED ELSE DROPPED.                       *
/. C  * -------------------------------------------------------------*
// SETPARM BPOR=NO                 BRAZIL       DICTIONARY
// SETPARM DANS=NO                 DANISH       DICTIONARY
// SETPARM DEUT=NO                 GERMAN       DICTIONARY
/. C
/. C  * -------------------------------------------------------------*
/. C  *                                                              *
/. C  * SELECT THE PART(S) YOU WANT TO PROCESS BY TYPING YES/NO      *
/. C  * -------------------------------------------------------------*
/. C  * SETPARM  | DESCRIPTION OF THE SETPARM PARAMETER              *
/. C  * PARAMETER|                                                   *
/. C  * -------------------------------------------------------------*
/. C  *   PART1  | BASIC INSTALLATION OF DM/VSE                      *
/. C  *          | 1.1 DELETES AND ADDS STANDARD LABELS FOR          *
/. C  *          |     THE VSAM FILES OF DM/VSE                      *
/. C  *          | 1.2 DELETES VSAM CLUSTERS OF DM/VSE AND           *
/. C  *          |     PREPARES MIGRATION                            *
/. C  *          | 1.3 DEFINES VSAM CLUSTERS FOR MASTER FILE AND     *
/. C  *          |     LOADS BASIC LANGUAGE CLIST DOCUMENTS          *
/. C  *          | 1.4 DEFINES AND LOADS THE RESTORE FILE            *
/. C  *          | 1.5 DEFINES AND INITIALIZES THE SAMPLE DATA SET   *
/. C  *          | 1.6 CREATES JOB TO CATALOG FCT MIGRATION ENTRIES  *
/. C  * -------------------------------------------------------------*
/. C  *   PART2  | DELETES, DEFINES, AND INITIALIZES                 *
/. C  *          | DW/370 WORKING CLUSTERS                           *
/. C  * -------------------------------------------------------------*
/. C  *   PART3  | LOADS LANGUAGE DOCUMENTS                         *
/. C  * -------------------------------------------------------------*
/. C  *   PART4  | PREPARES AND LOADS LANGUAGE DICTIONARIES          *
/. C  *          | FROM TAPE DM/VSE120 BASE2OF3                      *
/. C  * -------------------------------------------------------------*
```

Figure 55 (Part 4 of 5). Job for Customizing

```
/. C  *   PART5  | DEFINES AND LOADS FONT TABLES                    *
/. C  * ----------------------------------------------------------*
/. C  *   PART6  | CREATES A JOB STREAM TO CATALOG THE FCT ENTRIES  *
/. C  *          | FOR SELECTED DICTIONARIES AND MIGRATION          *
/. C  * ----------------------------------------------------------*
/. C  *   PART7  | ASSEMBLES AND LINKS DW/370 PROVIDED CICS BMS MAPS*
/. C  * ----------------------------------------------------------*
/. C  *   PART8  | LINKS DM/VSE DISOSS MIGRATION PROGRAM IPZDISSV   *
/. C  *          | AND LOADS THE SAMPLE PANELS OF DM/VSE            *
/. C  * ----------------------------------------------------------*
/. C
// SETPARM PART1=YES             BASIC INSTALLATION
// SETPARM PART2=YES             DW/370 WORKING CLUSTER
// SETPARM PART3=YES             LANGUAGE DOCUMENTS
// SETPARM PART4=YES             LANGUAGE DICTIONARIES
// SETPARM PART5=YES             FONTS
// SETPARM PART6=YES             BUILD DM/VSE FCT ENTRIES
// SETPARM PART7=YES             ASSEMBLE AND LINK CICS BMS MAPS
// SETPARM PART8=YES             LINK DM/VSE DISOSS MIGRATION PROG
/. C
/. C  ************************************************************
/. C  ************************************************************
/. C  *                                                          *
/. C  *  END OF PARAMETER LIST     (USER SELECTION)              *
/. C  *                                                          *
/. C  ************************************************************
/. C  ************************************************************
/. C
/. C  ************************************************************
/. C  ************************************************************
/. C  *                                                          *
/. C  *  START OF THE INSTALLATION PROCEDURE                     *
/. C  *                                                          *
/. C  ************************************************************
/. C  ************************************************************
/. C
/. C  * ----------------------------------------------------------*
/. C  * SET DICTIONARY SETPARM STATEMENTS CORRESPONDING TO THE    *
/. C  * LANGUAGE SELECTION                                       *
/. C  * ----------------------------------------------------------*
IF BPORT = YES THEN
// SETPARM BPOR=YES
IF DANSK = YES THEN
// SETPARM DANS=YES
IF DEUTS = YES THEN
// SETPARM DEUT=YES
/. C  * ----------------------------------------------------------*
/. C  * CHECK SETPARM PARAMETERS:                                 *
/. C  * IF THE SETPARMS ARE NOT ASSIGNED TO VALID VALUES         *
/. C  * THE JOB IS TERMINATED AND A MESSAGE IS WRITTEN.          *
/. C  * ----------------------------------------------------------*
```

Figure 55 (Part 5 of 5). Job for Customizing

Customizing Sample

Part 6. Appendixes

Appendix A. Bibliography

IBM System Manuals for VSE/ESA

SC33-6702	System Upgrade and Service
SC33-6703	Planning
SC33-6704	Installation
SC33-6705	Administration
SC33-6706	Operation
SC33-6796	Messages and Codes, Volume 1
SC33-6798	Messages and Codes, Volume 2
SC33-6799	Messages and Codes, Volume 3
SC33-6708	Networking Support
SC33-6709	Programming and Workstation Guide
SC33-6710	Guide for Solving Problems
SC33-6711	Guide to System Functions
SC33-6712	Unattended Node Support
SC33-6713	System Control Statements
SC33-6714	Diagnosis Tools
SC33-6715	System Macros User's Guide
SC33-6716	System Macro Reference
SC33-6717	System Utilities
SC33-6721	Extended Addressability
SC33-6799	VSE/ESA Turbo Dispatcher Guide and Reference
SC33-6723	LANRES/VSE Guide and Reference
SC33-6731	VSE/VSAM Commands
SC33-6732	VSE/VSAM User's Guide and Application Programming
SC33-6733	VSE/POWER Administration and Operations
SC33-6734	VSE/POWER Remote Job Entry
SC33-6735	VSE/POWER Networking
SC33-6736	VSE/POWER Application Programming
SC33-6738	VSE/ICCF Administration and Operation
SC33-6739	VSE/ICCF User's Guide
SC33-6741	REXX/VSE User's Guide
SC33-6742	REXX/VSE Reference

Online Information

A complete set of VSE/ESA manuals can be found on the CDROM that is available with the system.

SK2T-0060 VSE/ESA Online Library Omnibus Edition VSE Collection

IBM National Language Support Manuals

The following IBM books provide detailed information on National Language Support:

IBM National Language Information and Design Guides:

- *Volume 1, Designing Enabled Products, Rules and Guidelines* SE09-8001. This manual describes rules and guidelines for designing products that are enabled for national languages.
- *Volume 2, National Language Support Reference Manual* SE09-8002. This manual provides general information about languages and countries in the left-to-right, double-byte character set, and bidirectional groups.
- *Volume 3, Arabic Script Languages* SE09-8003. This manuals contains general information about Arabic script languages and some conventions on language usage.
- *Volume 4, Hebrew*, SE09-8004. This manual contains general information about Hebrew and some conventions on language usage.

Other IBM publications that are related to national language support:

- *DBCS Applications Primer (Enabling your programs for Chinese/Japanese/Korean)* GG18-9059. This manual contains guidance to write application systems with DBCS, in utilizing IBM software products with DBCS capability in the System/370 MVS environment.
- *DBCS Design Guide - System/370 Software* GG18-9095. This manuals contains information on what is required to write programs that are capable to handle DBCS characters such as Chinese, Japanese, or Korean, and to translate messages and panels issued by applications programs into such languages.
- *SAA CUA: Common User Access, Basic Interface Design Guide* SC26-4583. This manual defines SAA user interfaces for System/370 and AS/400 terminals for designer and developers of applications for nonprogrammable terminals.
- *SAA CUA: Common User Access, Basic Interface Design Guide 1991 Addenda* GG22-9508. This document describes migration considerations for applications written to conform to the SAA CUA architecture documented in the *SAA CUA Basic Interface Design Guide*
- *Character Data Representation Architecture Overview* GC09-1392. This manual introduces and explains Character Data Representation Architecture at a high level.
- *Character Data Representation Architecture Reference* Addenda, SC09-1390. This manual describes Character Representation Architecture in detail.
- *Character Data Representation Architecture Registry* GC09-1391. This manual for Character Data Representation Architecture lists the default conversions for character graphics.
- *ITSC - Keys to Sort and Search for Culturally Expected Results*, GG24-3516. This manual explains how to perform sorts and searches on sorted data in a manner that matches the expectations of end users. This manual concentrates on the Latin script languages of Western Europe. It also includes information on Arabic.
- *3174 Establishment Controller, Character Set Reference* GA27-3831. This manual provides a wealth of NLS information on the 3174 controller. It includes sketches of keyboards (with function keys), code pages, and character generators supported by the 3174 in the USA and in World Trade countries
- *3270 Data Stream: Programmer's Reference*, GA23-0059. This manual describes 3270 data stream information to support SAA.

Index

J

L

M

N

O

P

R

S

T

V

Z

Communicating Your Comments to IBM

IBM VSE/Enterprise Systems Architecture
Preparing a Product for VSE
Version 2 Release 4

Publication No. SC33-6331-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

  IBM Deutschland Entwicklung GmbH
  Department 3248
  Schoenaicher Strasse 220
  D-71032 Boeblingen
  Federal Republic of Germany

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456
- If you prefer to send comments electronically, use this network ID:

  IBM Mail Exchange: DEIBMBM9 at IBMMAIL

  INTERNET: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

IBM VSE/Enterprise Systems Architecture
Preparing a Product for VSE
Version 2 Release 4

Publication No. SC33-6331-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

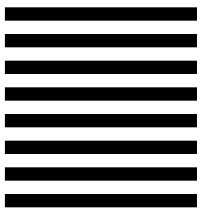
Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-6331-01

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Attn: Dept EHJ - BP/003D
6300 Diagonal Highway
Boulder, CO 80301-9151

Fold and Tape **Please do not staple** Fold and Tape

SC33-6331-01

Cut or Fold
Along Line

File Number: S370/S390-20
Program Number: 5690-VSE

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6331-01