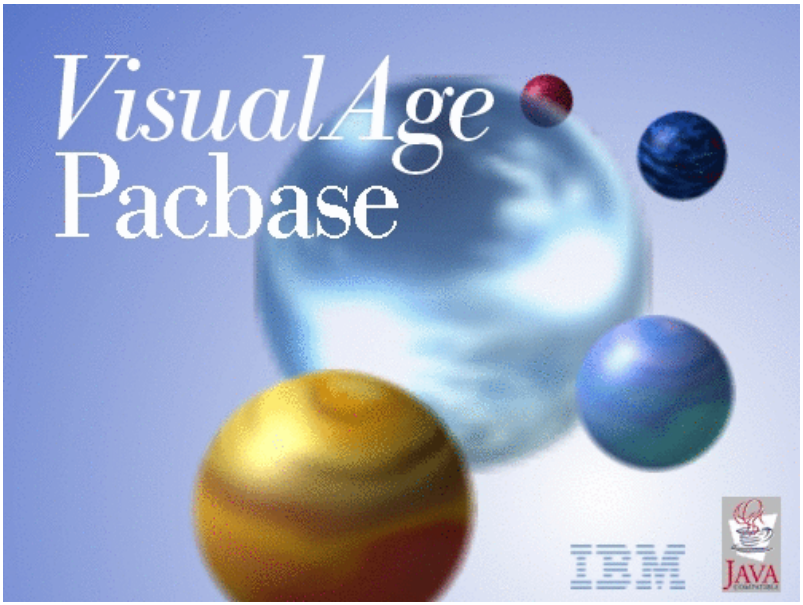
VisualAge Pacbase

DSMS: Installation & Operations Windows 2000 or NT Server

Version 3.5



VisualAge Pacbase

DSMS: Installation & Operations Windows 2000 or NT Server

Version 3.5

Note

Before using this document, read the general information under “Notices” on page v.

You may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

http://www.ibm.com/support/docview.wss?rs=37&uid=swg27005477

Consult the Catalog section in the Documentation home page to make sure you have the most recent edition of this document.

First Edition (April 2007)

This edition applies to the following licensed programs:
- VisualAge Pacbase Version 3.5

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at: http://www.ibm.com/software/awdtools/vapacbase/support.html or to the following postal address:

IBM Paris Laboratory
1, place Jean–Baptiste Clément
93881 Noisy-le-Grand, France.

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

Contents

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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service 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.

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 Paris Laboratory, SMC Department, 1 place J.B.Clément, 93881 Noisy-Le-Grand Cedex. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

Trademarks

IBM is a trademark of International Business Machines Corporation, Inc. AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

Chapter 1. Foreword

Use of the manual

This manual is intended for the person in charge of the installation and for the DSMS Database Manager.

It describes the DSMS components, the environment, the batch procedures, the instructions for installing the new version and the operations to be carried out for a standard reinstallation of corrected versions.

Note

DSMS 3.5 requires a complete installation of the technical package, i.e. files, programs and batch procedures.

Chapter 2. DSMS Components

Introduction

DSMS manages permanent data in batch and on-line mode.

The following types of resources are required to operate DSMS:

- Directories in which the DSMS operating programs and system parameters are stored.
- Permanent files which contain data manipulated by the DSMS function:
  - A system file containing error messages and HELP documentation on DSMS,
  - User files containing the User and Administrator data.

Note:

This manual describes the installation and operation of DSMS. DSMS can be installed independently of other VisualAge Pacbase functions and facilities.

For further details on the operation of the Function itself, refer to the DSMS Reference Manual.

'System' files

These are the files which make up the system itself. They are not affected by everyday operations and should be reloaded upon each re-installation.

These files are:

- the batch and on-line executable programs (directory: 'SYS\PGM')

as well as:

- the DE file, which contains the DSMS error messages and automatic documentation:

Characteristics	Value
Size	About 30,000 records (about 3.5 Mb)
Organization	Indexed
Length	90
Key	17 (position 1)
Location	SYS\SKEL directory

Characteristics	Value
Internal name	PACDDE

- the DH file (internal name PACDHE) is used to save the screen when the Help function is called in on-line mode.

  This file is located in the DATA\'db_name'\BASE directory ('db' = database).

'User' files

These files contain the user data which is managed by DSMS.

The first five files contain data that is directly managed by the system. These are:

- The DSMS data file (DA)

Characteristics	Value
Organization	Indexed
Length	Minimum 80, maximum 350
Key	40 (position 3)
Location	DATA\'db_name'\BASE directory
Internal name	PACDDA

- The cross-reference file (DX)

Characteristics	Value
Organization	Indexed
Length	80
Key	50 (position 1)
Location	DATA\'db_name'\BASE directory
Internal name	PACDDX

- The VA Pac Element file (DC)

Characteristics	Value
Organization	Indexed
Length	Minimum 50, maximum 168
Key	31 (position 3)
Location	DATA\'db_name'\BASE directory
Internal name	PACDDC

- The DSMS Journal file (DJ)

Characteristics	Value
Organization	Relative
Length	180
Location	DATA\'db_name'\JOURNAL directory
Internal name	PACDDJ

- The DAF TP work file (SYSDAF)

Characteristics	Value
Organization	Indexed
Length	Minimum 100, maximum 554
Key	37 (position 2)
Location	Chosen by the user
Internal name	PACDDF

Three other sequential files are used for the DSMS backup. These are:

- The Backup file (BB)

Characteristics	Value
Organization	Variable sequential
Length	354
Location	DATA\'db_name'\SAVE directory
Internal name	PACDBB

- The Journal Archive file (BJ)

Characteristics	Value
Organization	Sequential
Length	180
Location	DATA\'db_name'\SAVE directory
Internal name	PACDBJ

- The Deactivated Archive file (BQ)

Characteristics	Value
Organization	Sequential

Characteristics	Value
Internal name	PACDBQ
Location	None by default (non-assigned file)

User control sub-programs

The sources of user control sub-programs for the definitions of changes, events, sites, requests and layouts, as well as the DAF tables Dictionary (DAFDIC) can be downloaded through the VA Pac Support web page at

http://www.ibm.com/software/awdtools/vapacbase/support.html

Member	Contents
BVPCUAM	Online control on change definition
BVPCUEV	Online control on event definition
BVPCUMQ	Online control on layout definition
BVPCURQ	Online control on request definition
BVPCUSI	Online control on site definition
BVPDSCAM	Batch control on change definition
BVPDSCEV	Batch control on event definition
BVPDSCMQ	Batch control on layout definition
BVPDSCRQ	Batch control on request definition
BVPDSCSI	Batch control on site definition

Chapter 3. Installation

Prerequisite

Hordware and software

- Architecture: a Windows/NT, Windows 2000, Windows XP or Windows 2003 server.
- Processor: workstation with Windows NT server (4.0 minimum), Windows 2000, Windows XP or Windows 2003.
- Memory: 96 Mb. Additional memory may be needed depending on the number of servers installed on the same machine.
- Software:
  - Microsoft Windows Script, version 5.1 onwards. You can download it from the following URL:

    `www.microsoft.com/downloads`
- For the MICROFOCUS version:

  MICRO FOCUS Application Server
- For the Acucorp version:

  ACUCOBOL-GT

  CAUTION:

  Besides installing the cobol runtime, you must update the system environment variables (PATH, COBPATH...).

  For AcuCobol, the PATH system variable must be completed with the path of the AcuCobol runtime (usually ..\AcuGT\bin). The machine must then be rebooted.
- Installation medium : CD-ROM drive.

Disk space

The disk space required for the files varies according to the number and size of the applications managed by the system.

The disk space required to install the servers is 17 million bytes approximately.

Installation

Overall presentation

The product is supplied on a CD-Rom.

Administrator's rights are required to perform the installation.

You can change the disk drive and the name of the repository during the installation. By default, the server is installed in the root directory:

C:\Program Files\IBM\VisualAge Pacbase 35\

DSMS is installed in a special directory named \DSMS. The installation involves the following steps:

1. Installation of the system, which creates the \SYS\ directory
2. Installation of a DSMS Database, which creates the \CONFIG\ [db_name]\ and the \DATA\[db_name]\ directories
3. Restoration of the test Database via the DRST procedure.

Steps 1 and 2 can be executed independently or in a sequence, but always in this order.

On the other hand, the test Database is automatically restored right after is is installed.

All the Databases are created with the data which comes from the installation tests and configuration parameters.

CAUTION:

In this manual, the [db_name], which is used to designate the full directory name, is the name of the directory in which the Database data is stored. It is not the Database code.

System installation

The execution of setup.exe opens a graphical interface which guides you through the installation process.

After a Welcome screen, some workstation characteristics (which can be modified) are displayed:

- Owner's Name and Organization,
- Location for the installation

If a VisualAge Pacbase component is already installed on the server, its directory is used for the DSMS installation.

The installation process copies:

- the conversion file, the file which contains the error or information messages output by the procedures,
- the programs,
- the procedures,
- the Database creation utility,
- the startup script models,
- the error message files.

Repository installation

DSMS Database

You can install a DSMS Database directly after the system installation or independently, via the 'Start' menu where you can access the shortcut (Create New Database) of the [VisualAge Pacbase 3.5 DSMS Server] group.

To install a DSMS Database, you must specify the following parameters:

- the Database name (8 char.)
- the Database code (4 char.),
- the Database port number (between 49152 and 65535, unique for each Database).
- the Database language code (English by default).

More than one DSMS Database can be installed. Each has its own environment.

You can modify the proposed directories (C:\Program Files\IBM\VisualAge Pacbase 35\).

You can modify the backup directory, the temporary file directory, or the product user directory in which the procedures' execution reports are strored, among others.

After this installation, the test Database supplied is restored.

Configuration parameters

For each Database, a Configuration directory is created with

1- BvpServer.ini, which contains the parameters of the conversational server,

Complement: installation of DAF environment

DSMS ACCESS FACILITY (DAF) implies the transformation of the DSMS Database access SQL queries, written in user programs, via the generation of data and calls to sub-program in the generated COBOL source of these programs.

So the pre-processor processes the programs generated by VA Pac to make this transformation.

The pre-processor is constituted of a program, BVDAFD10.exe, installed in the SYS\PGM directory.

You can use the DPDF procedure to process your generated programs using DAF (see chapter dedicated to the DPDF procedure).

The work file required to operate DAF is described in paragraph 'DAF TP Work file' in subchapter 'The User Files', chapter 'The DSMS Components'.

Extraction sub-programs

For user programs generated with a variant 3 (adaptation to COBOL Micro Focus), the same extraction sub-programs are used for batch and on-line programs. These extractors are compiled linked when delivered (.gnt files) in the SYS\PGM directory.

The extraction programs are:

- BVPDSBDF and BVPDTPDF, called by the DAF user programs.
- BVPDSDAC and BVPDSFAC, called by the extractors to access the DSMS Database and the DAF work file.

DAF Dictionary

The Data Element, Data Structure and Segment entities can be used to write programs which the DAF facility. These entities are delivered as batch transactions in the DAFDIC file which you can download from the VisualAge Pacbase Support URL:

http://www.ibm.com/software/awdtools/vapacbase/support.html

The insertion of the 'DAF dictionary' into VisualAge Pacbase via the UPDT batch update procedure is under the responsibility of the Database Manager who must first: check the compatibility of entity codes with the entities already existing on the site.

In order to avoid problems of compatibility between the site dictionary and the entities provided for the DAF facility, it is recommended to create an independent library nework for the writing of the site's DAF utilities.

Compiling and running DAF prorams

The compiler used for DSMS programs is Micro Focus version 4.0.nn. It is advised to use the same level of compiler in order to prevent conflicts between Micro Focus libraries.

Compile instructions (COBOL.DIR file):

```
ASSIGN "EXTERNAL"
SEQUENTIAL "LINE"
```

Compile and link command:

```
CBLLINK -E DAFPGM.CBL
```

Exeuting a DAF Extractor

The following files must be assigned before the DAF extractor is executed:

- Permanent input files:
  - DSMS data file : PACDDA
  - VA Pac element file : PACDDC
  - Cross-reference files : PACDDX
  - Error message file : PACDDE
- DAF work file : SYSDAF
- User files if relevant.

List of installed elements

The installation process creates the following directories:

- Dsms\CONFIG.

  This directory contains the Bvpserver.ini file in a 'db_name' sub-directory for each DSMS Database installed.
- Dsms\DATA.

  This directory contains a 'db_name' sub-directory for each DSMS Database installed. All the Database data is stored there (user files, temporary files, procedures' execution scripts and reports).
- Dsms\SYS.

  This directory is common to all the DSMS Database installed and contains all the load-modules of the DSMS function, the system files, the procedures and various facilities.

Putting a Va Pac Database under DSMS control

Implementation under VisualAge Pacbase

You put a VA Pac Database under DSMS control via the Security browser of the Administration Database.

In the 'DSMS control' tab, you must associate the DSMS transaction code ('DSMS Database code') with the code of the selected Database.

You can associate the same DSMS Database code with more than one VA Pac Database, or one DSMS Database code with only one VA Pac Database.

An 'Administrator' profile is required for this operation.

Implementation under DSMS

The screen accessed through the PL choice enables you to specify, for each VA Pac Database, which libraries, sessions and entities are to be controlled by DSMS.

Note: In this screen, the VA Pac Database code that is required is the Database logical code, displayed in the top right-hand corner of the VA Pac screens. This code can be modified by a user input in the REST procedure (restoration procedure).

For more information, refer to the DSMS reference manual, Chapter 'VA Pac interface: Database Lock'.

One VA Pac Database and one DSMS Database

If a VA Pac Database is monitored by a DSMS Database, the implementation described above is quite adapted. The standard installation of DSMS and VA Pac locates the DSMS DC file under the DATA\'db_name'\ directory of the VA Pac installation. It thus allows the control of the VA Pac Database if relevant.

However, if a VA Pac Database is to be monitored by a DSMS Database, the location of the DSMS Database DC file must be modified. The file must be moved to the DATA\'db_name'\BASE directory of the VA Pac Database to be monitored.

To relocate DC, change the assignment of the DC file of both the DSMS and VA Pac installations in the Server.wsf procedure. This procedure is in the SYS\PROC directory of the DSMS and VA Pac installations.

Multiple VA Pac and DSMS Databases

If several VA Pac Databases are monitored by only one DSMS Database, the adequate implementation procedure is the one described above. But the standard DSMS and VA Pac installations require the DSMS DC file to be located in the DATA\'db_name'\BASE directory of the VA Pac installation. It can then monitor all the VA Pac Databases if necessary.

However, if several VA Pac Databases are to be monitored by several DSMS Databases, the implementation steps described above are not sufficient. Each DSMS Database's DC file must be relocated to the DATA\'db_name'\BASE directory of the VA Pac Database to be monitored.

To relocate DC, change the assignment of the DC file of both the DSMS and VA Pac installations in the Server.wsf procedure. This procedure is in the SYS\PROC directory of the DSMS and VA Pac installations.

Utilization tests

These tests break down into the following phases:

- On-line use tests,
- Extraction utility test,
- Database management tests.

1. ON-LINE USE TESTS

Start up an on-line server.

Connect a workstation to it: the user codes defined in the test Database are TEST or USER, the password is IBM (French is the TEST language and English is the USER language).

Work in the DSMS Database, in read-only and then read-write mode.

2. EXTRACTION TEST

Run the DEXT procedure which extracts elements from the test Database.

For this test, the on-line server may remain active.

3. DATABASE MANAGEMENT TESTS

The objective of these tests is to run the Database management procedures.

These tests consist of the following steps (to be performed in this order):

- Archive the journal created during the tests: run the DARC procedure which outputs the BJ.NEW file, then run the BJBACKUP command file at the end of the archiving to create the BJ file (and the BJ-1 file if necessary).

- Back up the Database directly: run the DSAV procedure which outputs the BB.NEW file, then run the BBBACKUP command file at the end of the backup to create the BB file (and the BB-1 file if necessary).
- Restore the Database from the BJ archive and the BB Database backup: run the DRST procedure.

For all these tests, the on-line servers must be shut down.

After restoring the Database, perform another set of quick operational on-line tests (but make sure to start up the on-line server first).

Connection

Starting up and closing down a server

The Listener must be installed in Service NT mode.

Workstations can then connect to DSMS.

The 'Start [db_name] Database Service' shortcut located under the [VisualAge Pacbase 3.5 Dsms Server] program group can be used to start up the listener on the [db_name] Database.

The 'Stop [db_name] Database Service' shortcut located under the [VisualAge Pacbase 3.5 Dsms Server] program group can be used to close down the listener on the [db_name] Database.

The operating parameters of the listener are defined in the 'Server.wsf' procedure.

When abnormal executions occur in a listener, messages can be displayed in the list of events of the 'Event Viewer' administration tool, under 'Log/Application'.

These 'ERROR'-category events can give a first indication. They generally correspond to problems in the execution environment.

However, in case of abnormal processing, the product support may ask you to activate the 'trace mode' to detect the source of the error.

- TRACE MODE

Different trace levels can be implemented:

- Level 1

  Minimum trace allowing to follow the listener processing with the calls to the COBOL communication monitor,

- Level 2

  Detailed trace of the listener processing,
- Level 4

  Trace of the messages between the listener and the client workstation.

  In the listener startup procedure, 'server.wsf', the purpose of the SRV_TRACE environment variable is to activate the trace mode. To use another trace level, you must set the SRV_TRACE variable and re-start the listener.

  EXAMPLE:

  SRV_TRACE=1 for a trace level 1

  SRV_TRACE=3 for a trace level 1 and 2

  SRV_TRACE=5 for a trace level 1 and 4

  There are two types of trace result files:

  - srv[process_number].txt

  to trace the listener (BvpServer.exe).

  - dial[process_number].txt

  to trace each listener connection (BvpDial.exe).

  These files are located in the SRV_DIR directory, set by default in server.wsf file to:

  .../data/[database_name]/tmp

  The SRV_TRACE_DEL environment variable is set to keep all or a part of the traces created by the listener execution in "dialnnn.txt" :

  SRV_TRACE_DEL : "ON" (default value)

  to keep traces produced by a processing error only.

  SRV_TRACE_DEL : "OFF"

  to keep all the traces.

Connecting to a 3270 emulator

You can access an on-line server, in character mode, via a 3270 emulator.

To access a database in a 3270 mode via an on-line server, you must configure the emulator by indicating:

- the IP address of the machine where the on-line server is implemented,

- the on-line server listening port number, chosen upon installation when the Database is created.

The code page of the emulator must be valorized according to the database language code:

- code page 1147 for a French Database,

- code page 1146 for an English Database.

These code pages are automatically set in the 'Server.wsf' procedure when the online server is started up.

Transaction code of the user parameters: 'TABPARM'.

Transaction code for the access to the Pactables function: 'PACTABLES'.

Chapter 4. Reinstallation of the components

The system environment of the DSMS server must be re-installed whenever enhancements to the currently installed version are being delivered.

The Windows NT service -Add/remove programs- enables you to do it with the 'Add/remove' options.

Reinstallation

The system update is automatically proposed if you have an older version already installed.

The operational startup scripts, in the \DATA\[Database_name]\SCRIPT directories, are not impacted. To get the last version of the scripts, upon the re-installation, refer to the startup script models in the \SYS\Script_Models directory.

Uninstallation

You uninstall the system environment via the 'Add/remove programs' service of the control panel ('VisualAge Pacbase DSMS Server' application).

At the same time, you are given the possibility to remove all data accesses from the Registry while saving this data in another \DATA directory.

Deletion of data

Deleting a Database is an operation that cannot be undone. It is then advised to save the Database first.

The deletion is performed via manual operations:

- Delete the directory which contains the Database data, and all its dependent sub-directories,
- Delete the directory:

```
Winnt\Profiles\AllUsers\Start Menu\Programs\VisualAge
Pacbase 3.5 DSMS Server\[database_name]
```

- Delete the key from the HK_Local_Machine registry:

```
Software\IBM\BVP VisualAge Pacbase 3.5\DsmsBVP_DATA\[database_name].
```

Chapter 5. Retrieval of a 2.n version

Overall presentation

To retrieve a 2.n version, you must perform the following operations:

- save the 2.0 version,
- reorganize the 2.0 version,
- restore in the new version's environment, using the file produced by the previous reorganization.

Chapter 6. Batch procedures

Introduction

The batch processing associated with DSMS is divided into procedures. The following chapters describe each of these procedures that may be used and give details on its specific execution conditions.

For each procedure, you will find:

- A general presentation containing:
  - an introduction,
  - the execution condition(s),
  - the action(s) to be taken in case of abnormal execution,
- The description of user input, processing, results, and possible recommendations on use.
- A description of each step containing:
  - The files used (temporary and permanent),
  - The return codes that may be generated by each step.

Classification of procedures

There are various types of batch procedures.

DATABASE MANAGEMENT PROCEDURES:

- Initialization of DSMS files (DINI)
- Archiving of file update transactions (DARC)
- Restoration of files using the backup and archived files (DRST)
- Backup of files (DSAV)
- Reorganization of cross-references files (DREO).

UTILITY PROCEDURES:

- Extraction, from the VA Pac Journal, of the transactions which correspond to the modified VA Pac entities related to Changes (DEXP).
- Extraction, from the DSMS journal (DXBJ), of the transactions for the DUPT batch update.
- Printing of query results, and of table and keyword lists requests (DPRT).
- Extraction, from DSMS, of Events, Changes, Sites or Tables as batch transactions (DEXT).

- Batch update of DSMS files of Events, Changes, Sites or Tables (DUPT, DUPD).
- Pre-processing of DAF source files (DPDF).
- Renaming of Table, Site and Keyword codes (DREN).

RETRIEVAL OF A DATABASE ON ANOTHER PLATFORM:

- Replacement of low-values with blanks (DLVB).

Abnormal endings

Abends may occur during the execution of a batch program. Input-output errors on the system or Database files cause a forced abnormal end with a return code '12', described by a message in the .Log file of the procedure.

When an abend occurs, you must find the error message. This message is displayed in the following format:

```
PROGR : pppppp  INPUT-OUTPUT ERROR : FILE ff  OP : oo
STATUS : ss
END OF RUN DUE TO PROVOKED ABEND
```

This message appears if you have previously set the 'NoDisp' variable to 'NO' in Incl.vbs.

In most cases, examining the status and type of operation enables you to find the cause of the abnormal execution.

The summary table below lists the most common values for the status and type of operation.

Code	Operation
W	WRITE
RW	REWRITE
RU	READ UPDATE
OP	OPEN
CL	CLOSE
D	DELETE
R	READ
P	START
RN	READ NEXT

Status	Message
10	End of file
21	Sequence error
22	Duplicate key
23	Record not found
24	Boundary violation
30	System error
34	Boundary violation (sequential)
92	Logical error (For example, the opening of a file which is already open)
93	File still open in on-line mode
95	Invalid or Incomplete file

When there is no such message, and if the type of ABEND generated directly reports a problem in the product programs, contact the product support at IBM. KEEP ALL LISTINGS that may be necessary to analyze the problem.

If the error is not an input-output error on a Database file, the following message is displayed:

```
Run Time Error nnn
```

where nnn is the error number.

The Run Time Error 013 is the most frequent. It indicates that the procedure did not find an input file.

The next subchapter contains the list of the most frequent errors. Each Run Time Error is briefly described.

If the Run Time Error is not in the following list or if its associated description is not explicit enough and if the error directly involves the system programs, you must contact the Hot Line and keep all listings which might be useful in solving the problem.

List of 'runtime errors'

This list is a reminder of the most common errors and their meaning.

```
Number   Meaning
------   -------

004      Invalid file name
005      Invalid device specification
007      No more disk space
009      Directory full or does not exist
```

```
013      File not found
026      Block I-O error
027      Device not available
028      Disk space exhausted
033      Physical I-O error
105      Memory allocation error
116      Cannot allocate memory
135      File not found
150      Program abandoned on user request
157      Not enough program memory:  object file too big
         to load
170      System program not found
173      Called program file not found
188      File name too long
198      Not enough program memory: object file too large
         to load
207      Machine does not exist on the network
208      Network communication error
209      Network communication error
221 !
222 !>   Error during a SORT
223 !
```

Management of errors in the procedures

If an error is detected in a step, the next steps are not executed. The name of the erroneous program and, if possible, the type of the detected error, are displayed.

The procedure then displays the message:

```
"Press Return to carry on"
```

You must then stop the procedure, in order to view the error if several procedures follow one another.

(If the NOBVPERR environment variable is set to 'yes', this message is not displayed and you do not have to stop the procedure)

The procedure stops with a return code other than zero. This code can be retrieved via the Return variable right after the command which submits the procedure. This prevents the execution of the next procedures if various procedures are executed in sequence.

Procedure definition

A procedure is a Windows Scripting script (.wsf), which includes Visual Basic Script scripts (.vbs).

Each procedure can be executed only via a launch Script (.wsf) which contains the following information:

- a <resource id> field constituted of input data (141 lines maximum) or the definition of an input file by WshEnv(″BVP_Input″) = ″full file name″ (priority value),
- the user code,
- the Database external name (directory under \DATA),
- all the assignments of the procedure files that can be parameterized.

  Example: WshEnv(″PDS900_PAC7DGY″) = ″External table name″.

You must enter all this information before launching the procedure.

Other elements can also be specified to assign files or parameterize procedures. (e.g. whether the message must be displayed or written in a file).

In any case, the provided user input must be verified and adapted to your environment (<resource id> or BVP_Input).

The temporary files, execution reports and output files are found under directories created dynamically.

- Temporary files are located under:

  \DATA\[db_name]\TMP\[user_code]\[proc_name]-[number]

  They are deleted at the end of the procedure execution. If you do not want these files to be deleted, you must previously set the ″DelDo″ variable to ″NO″ in Incl.vbs. So you will have to delete them manually.

  Once this variable is set (″NO″ or ″YES″), it applies to ALL the procedures.
- Output files, execution reports are located under:

  \DATA\[database_name]\USERS\[user_code]\ [proc_name]-[number]

  This number is an application execution number ; by default it is the process number of the procedure.

  CAUTION:

  In DSMS for Acucorp, when a procedure containing a Cobol sort abends, the temporary sort files may not be deleted automatically. These files (16 in total, with names which begin with ’T’) are stored in the WINDOWS temporary directory (\TMP or \TEMP) or directly under ’C:\’. Their date-time correspond to the date and time of the procedure startup.

Procedure submission

You can execute the launch script (BVPproc.wsf) or the supervisor (PRBVP.vbs):

- via a command line,
- by double-clicking on them,

- via the 'Start' menu. Access the shortcut: [[dbase_name] Database Utilities] in the [VisualAge Pacbase Server] group of [dbase_name] and enter [Script_name].

NB: The Prbvp.vbs supervisor is an independent VBScript procedure which starts the execution of the launch scripts. It requires 2 arguments: [dbase_name] [script_name] ; [dbase_name] is the directory under \DATA.

To be executed, each procedure (whose name is represented by xxxx) requires the definition of various elements:

1. either in the launch script: This launch script, BVPxxxx.wsf, is in Windows Scripting.
   a. User code,
   b. Database external name (directory under \DATA),
   c. The input data is described in the <resource id> field of the script and retrieved in the BVP_Resource environment variable (in this case, the data volume is limited to 141 lines), or can also be contained in a file named in the BVP_Input environment variable (in this case, the data volume is not limited).
   d. You can assign files, other than the input file, 'in substitution' by defining an environment variable with:

      . 'StepName_FileName' for a substituted assignment in a particular step,

      . or 'ProcName_FileName' for a substituted assignment in the whole procedure.

      For example, you define the variable WshEnv('PDS900_PAC7GY') = Fil_In with pathname to indicate a substituted assignment in the PDS900 step of the DUPD procedure.
   e. Information or error messages are:

      . either displayed on the screen (MsgTyp = 1),

      . or edited in a file (MsgTyp =2). This is the default value.

      You parameterize this in the BVP_Msg environment variable.

CAUTION: DSMS does not manage errors and missing elements in the startup script.

1. or in the part which is common to all procedures. This common part, named 'incl.vbs', is a VBScript inclusion, in each procedure.
   a. Retrieval of the arguments: [dbase] and [user]
   b. Assignment of 2 parameters with 2 possible values:

      DelDo = 'YES'/'NO': the temporary files are deleted (YES) or not (NO).

      NoDisp = 'YES'/'NO':

The cobol 'displays' are not displayed (YES) or are displayed by cmd (NO).

c. Constitution of the MB file (input data) which contains the user code, password, sometimes a command and analysis elements.

   If the [user] user code is not found in the launch script, it is searched for in the input file.

   If the elements required for the execution of the procedure are missing, an error occurs.

Chapter 7. DARC - Journal archiving

DARC - Introduction

The Journal Archiving procedure (DARC) backs up the Journal file (DJ) as a sequential file (BJ), and reinitializes it both logically and physically.

The new archived transactions do not overwrite the transactions previously archived; they are added to them.

The previously archived transactions can be deactivated, if requested.

Execution condition

The database must be closed to on-line use.

Even if the actual closing of on-line access is not controlled by the procedure, it prevents any other update while the procedure is being executed.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

If the abnormal end occurs before the step which creates the Journal file (DJ), the procedure can be restarted as it is, after the problem has been solved.

If the abnormal end occurs during or after this step, the user input must be modified before a new execution of the procedure so as to specify a reinitialization request without a backup of the Journal file (already backed-up).

DARC - Input / Processing / Results

USER INPUT

The DARC procedure includes an optional input to:

- deactivate the previously archived transactions that are now obsolete,
- indicate the absence of previously archived transactions as input,
- indicate the unavailability of the Data file (DA) as input,
- request only a reinitialization of the transaction file.

The structure of this input is as follows:

Pos.	Len.	Value	Meaning
2	1	'S'	Line code
3	4	nnnn	Session number
7	8	CCYYMMDD	OR date up to which the user requests deactivation
15	1	'I'	Absence of previously archived transactions
16	1	'D'	Data file (DA) unavailable
17	1	'J'	Re-initialization without archiving

The session number and the date are exclusive. They are ignored if the absence of previously archived transactions has been indicated.

The unavailability of the Data file is to be indicated only when this file has been physically deleted (see paragraph 'Recommendations').

The request for a reinitialization without archiving is necessary when the Journal file is lost physically.

Caution:

In this case, the previous archiving is not duplicated on the output archiving. When the cataloging is automatic, previous archiving may be lost if no uncataloging is performed.

In case of an error on one of the options, an error message is printed and the archiving is generated using the default options.

Recommendations

If there is no user input, this procedure can be executed only if the database is in a consistent state, and if the Journal file is correctly formatted.

When data needs to be restored after a problem, some information in the database may be destroyed and neither the DARC nor the DRST procedures can then be executed.

In this case, AND IN THIS CASE ONLY, columns 15 to 17 of the user input must be used as follows:

- If the Data file (DA) is lost or has been flagged as 'inconsistent', a 'D' in column 16 means that the DARC procedure will not take the Data file (DA)

into account. However, the DRST procedure must be executed afterwards, since under these conditions, the DARC procedure makes the DA data inconsistent.

- If the Journal file (DJ) is lost or destroyed, a 'J' must be entered in column 17. The DARC procedure formats an empty Journal file. The DRST procedure can then be executed.
- If the sequential Archived file (BJ) is lost or destroyed, an 'I' must be entered in column 15. The DARC procedure will format a new sequential archive file.

If one of these columns is accidentally set to its value, and the DARC procedure executed when the Data (DA) file is in a consistent state, the consequences are :

- 'I' in col. 15: The transactions previously archived are lost. All the transactions can be recovered by concatenating BJ(-1) and BJ(0) to obtain BJ(+1).
- 'D' in col. 16: The DARC procedure has to be re-run BEFORE any update. If it is done afterwards, the data is lost and a complete restoration must be executed.
- 'J' in col. 17 : The contents of the Journal file are lost and cannot be retrieved.

REPORT RESULTS

This procedure prints a report giving the number of archived update transactions and, if applicable, the number of records that have been deactivated.

GENERAL RESULTS

Once this procedure is executed, a sequential file containing all the archived transactions is produced.

The Journal file is re-initialized.

It is also possible to store in another file all update transactions that have been deactivated.

Note: This procedure does not increment the current session number of the database.

DARC - Description of steps

Archiving of journal file: PDS300

This step executes the following processing:

- Updates the file of archived update transactions,
- Positions a flag in the Data file which represents the journal archiving,
- Writes the deactivated transactions onto a special file, if deactivation is requested by user input.

Code	Physical name	Type	Label
PACDMB		Input	User transaction
PACDJB	Save dir.: BJ	Input	Transactions previously archived
PACDDJ	Journal dir.: DJ	Input	Journal file to be reinitialized
PACDDE	System - skel. dir.: DE	Input	Error message file
PACDDA	DBase dir.:DA	Input Output	Data file
PACDBJ	Save dir.: BJ.NEW	Output	Updated archived transactions
PACDBQ	To be assigned in order to keep deactivated trans.	Output	Deactivated archived transactions
PACDRU	User dir.: DARCRU300.txt	Report	Archiving report

Return codes::

- 0 : No error detected on the files.
- 8 : User Input error.
- 12 : Input-output error on a file.

Re-initialization of the journal file: PDS320

This step executes the following:

- Creates a record in the Journal file
- Repositions the Data file flag.

Code	Physical name	Type	Label
PACDMB		Input	User transaction
PACDDE	System - Skel. dir;: DE	Input	Error-message file
PACDDA	DBase dir.: DA	Input Output	Data file

Code	Physical name	Type	Label
PACDDJ	Journal dir.: DJ	Output	Journal file to be reinitialized
PACDRU	User dir.: DARCRU320.txt	Report	Reinitialization report

DARC - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'             - ARCHIVAL OF THE JOURNAL -
'
' ------------------------------------------------------
'
' INPUT       : COMMAND FOR DEACTIVATION OF ARCHIVED
'               TRANSACTION
' COL 2       : "S"
' COL 3 TO  6 : SESSION NUMBER
' COL 7 TO 14 : DATE (CCYYMMDD)
' COL 15      : " " PRESENCE OF ARCHIVED TRANSACTION FILE
'             : "I" ABSENCE OF ARCHIVED TRANSACTION FILE
' COL 16      : " " PRESENCE OF DATA FILE (DA)
'             : "D" ABSENCE OF DATA FILE (DA)
' COL 17      : " " ARCHIVAL AND REINITIALIZATION
'             : "J" REINITIALIZATION WITHOUT ARCHIVAL
'
' IN THE ABSENCE OF INPUT (OR ERROR ON A COMMAND PARAM.)
' NO DEACTIVATION WILL TAKE PLACE, HOWEVER ARCHIVAL AND
' REINITIALIZATION WILL BE EXECUTED NORMALLY.
'
' TRANSACTIONS WHOSE SESSION (DATE) IS PRIOR OR EQUAL TO
' THE SESSION (DATE) INDICATED ARE NOT KEPT. THEY ARE
' RECOVERED IN THE FILE OF DEACTIVATED TRANSACTION.
'
' ------------------------------------------------------
<job id=DARC>

<script language="VBScript">
Dim MyProc
MyProc = "DARC"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PDS300"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
```

```
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDMB") = Fic_Input
WshEnv("PACDJB") = Rep_SAVE & "\BJ"
WshEnv("PACDBJ") = Rep_SAVE & "\BJ-new"
WshEnv("PACDBQ") = Rep_TMP & "\NULBQ.tmp"
'PACDBQ not used
Call BvpEnv("PDS300","PACDRU",Rep_USR & "\DARCRU300.txt")
Call RunCmdLog ("BVPDS300")

If Return = 8 Then
Call Msg_Log (Array("1030"))
End If
If Return = 12 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PDS300")


Call Msg_Log (Array("1022" , "PDS320"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDS320","PACDRU",Rep_USR & "\DARCRU320.txt")
Call RunCmdLog ("BVPDS320")
Call Err_Cod(Return , 0 , "PDS320")

Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\BJ")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 8. DPRT - Printing of queries and output reports

DPRT - Introduction

The DPRT procedure performs all the printing operations for DSMS:

- Results of User Queries on Events, Changes and Sites, (this order must be respected)
- Printouts of Tables, Keywords, Queries and Reports.

See the DSMS Reference Manual for practical information on how to submit a DPRT execution in either batch or on-line mode.

Note: Printouts of Tables and Keywords can be submitted in batch mode only.

Technical information regarding the JOB Function (which enables you to submit the DPRT procedure in on-line mode) is given at the end of this chapter.

Execution conditions

None.

The Database can remain open to on-line processing.

Abnormal execution

Refer to Chapter 'The Batch Procedures', Subchapter 'Abnormal Execution'.

DPRT - Input / Processing / Results

USER INPUT

A '*' line (required):

Col.	Len.	Value	Description
2	1	'*'	Line Code
3	8	uuuuuuuu	DSMS User Code
11	8	pppppppp	Password
19	3	ppp	Product Code
22	2	su	Subsidiary Code
24	1	l	Language Code

4 report types exist, 1 line per printout is necessary :

- Tables

Col.	Len.	Value	Description
02	03	Txx	Codes of the Txx table
07	02	C1	... with their labels in the language of the connected user (default option)
07	02	C2	... with all their labels
02	03	TUD	User codes with all their authorizations (TUG + TUP + TUS)

- Queries / Reports

Col.	Len.	Value	Description
02	04	X QC	Query on Changes
		X QE	Query on Events
		X QS	Query on Sites
02	04	X RC	Report on Changes
		X RE	Report on Events
		X RS	Report on Sites
06	06	xxxxxx	Query or Report code
12	08	uuuuuuuu	User code for Query or Report owner (default value: connected user code)
20	02	C1	Printing of all description lines for the Query/Report type (default option)
		C2	Printing of only useful Query/Report description lines

- Lists

Col.	Len.	Value	Description
02	03	LJQ	Control cards
02	04	LCQC	Query on Changes
		LCQE	Query on Events
		LCQS	Query on Sites
02	04	LCRC	Reports on Changes
		LCRE	Reports on Events
		LCRS	Reports on Sites

Col.	Len.	Value	Description
07	02	C1	Printing of all description lines for the Query/Report type (default option)
		C2	Printing of only useful Query/Report description lines
12	08	uuuuuuuu	User code for Query/Report owner

- Keywords

Col.	Len.	Value	Description
02	04	LAKC	Stand-alone Keywords of Changes
		LPKC	Principal keywords of Changes
		LGKC	All the keywords of Changes
06	01	l	Keywords language code (default: connected user language code)
02	04	LAKE	Stand-alone Native Keywords of Events
		LPKE	Principal Native Keywords of Events
		LGKE	All Native Keywords of Events
02	04	LAKT	Stand-alone Techn. Keywords of Events
		LPKT	All principal keywords of Events
		LGKT	All the keywords of Events

Print via user query (99 queries maximum):

Col.	Len.	Value	Description
2	1	'Q'	
3	1	'C'	For a query on Changes
		'E'	For a query on Events
		'S'	For a query on Sites
5	6	rrrrrr	Code of the user Query (required) 'Q' Entity used.
5	6	mmmmmm	Code of the Report (optional)
17	1	d	Delimiter (optional)
			Parameters:
18	1	s	Symbol (optional)
19	1	x	Separator (optional)
20	54	.....	Parameter values (optional)

Col.	Len.	Value	Description
			If optional fields have not been filled in, default values are used. They come from the definition lines of the user Query found in the Database.

PRINTED OUTPUT

Two types of printed output are obtained:

- Results of user-defined Queries on Events, Changes and Sites.
- Standard printouts of Tables, Keywords, Queries and Reports.

Return code

Code	Description
0	OK with Queries
4	OK with tables, keywords, Queries/Reports print requests
8	OK with erroneous Queries or other requests
12	Fatal error
16	Sort error

DPRT - Description of steps

This procedure calls a unique program (PDSB) that acts as a flow monitor for the various programs, which are therefore sub-programs of this monitor.

The procedure includes the following steps:

The input file is automatically formatted when QUERIES are submitted in on-line mode.

Printing: PDSB

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDC	DBase dir.: DC	Input	VA Pac element file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDMB		Input	User Queries
PACDKD	Tmp dir.: WKD	Work file	Print requests
PACDKQ	Tmp dir.: WKQ	Work file	Queries

Code	Physical name	Type	Label
	Tmp dir.: W1 W2 W3 W5	Work file	Temporary files
			Temporary files
PACDQR		Work file	
			Temporary files
PACDQJ		Work file	
			Temporary files
PACDW1		Work file	
			Temporary files
PACDW2		Work file	
			Temporary files
PACDW3		Work file	
			Temporary files
PACDW4		Work file	
PACDIA	User dir.: DPRTDA.txt	Report	Flow report
PACDIB	User dir.: DPRTDB.txt	Report	List of Queries and requests
PACDID	User dir.: DPRTDD.txt	Report	Printing of tables and keywords
PACDIQ	User dir.: DPRTDQ.txt	Report	Report of extractions by Queries
PACDQI	User dir.: DPRTQI.txt	Report	Printing of extractions results
PACDRQ	Tmp dir.: DPRTRQ.txt	Report	Printing of Queries/Reports
PACDJQ	Tmp dir.: DPRTJQ.txt	Report	Printing of control cards

DPRT - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'             - PRINTOUTS AND QUERIES -
'
' ------------------------------------------------------
'
' INPUT :
' .. IDENTIFICATION LINE
'  COL 02     : *
'  COL 03     : DSMS USER CODE
'  COL 11     : PASSWORD
'  COL 19-21  : PRODUCT CODE
'  COL 22-23  : SUBSIDIARY CODE
'  COL 24     : LANGUAGE CODE
'
```

```
'  ..  EXTRACT COMMAND LINE(S)
'      ----------------------
'  COL 02-05  : TYPE OF EXTRACTION
'  -- EXTRACTION BY USER QUERY :
'  COL 05-10  : QUERY CODE
'  COL 17     : DELIMITER                    <--- OPTIONAL
'  COL 18     : SYMBOL                       <--- OPTIONAL
'  COL 19     : SEPARATOR                    <--- OPTIONAL
'  COL 20-73  : PARAMETERS VALUES            <--- OPTIONAL
' --- EXTRACTION OF QUERIES/LAYOUT :
'  COL 06-11  : QUERY OR LAYOUT CODE
'  COL 12-19  : OWNER OF THE QUERY/LAYOUT    <--- OPTIONAL
'
' ------------------------------------------------------
'
<job id=DPRT>

<script language="VBScript">
Dim MyProc
MyProc = "DPRT"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
CodLang = WshShell.RegRead (Rep_D & "\BaseLang")

 Call Msg_Log (Array("1022" , "PDSB" & CodLang))

'-------------------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDSB","PACDG6",Rep_USR & "\DPRTG6.txt")
Call BvpEnv("PDSB","PACDIB",Rep_USR & "\DPRTIB.txt")
Call BvpEnv("PDSB","PACDIA",Rep_USR & "\DPRTIA.txt")
Call BvpEnv("PDSB","PACDID",Rep_USR & "\DPRTID.txt")
Call BvpEnv("PDSB","PACDIQ",Rep_USR & "\DPRTIQ.txt")
Call BvpEnv("PDSB","PACDJQ",Rep_USR & "\DPRTJQ.txt")
Call BvpEnv("PDSB","PACDQJ",Rep_USR & "\DPRTQJ.txt")
Call BvpEnv("PDSB","PACDQI",Rep_USR & "\DPRTQI.txt")
Call BvpEnv("PDSB","PACDQR",Rep_USR & "\DPRTQR.txt")
Call BvpEnv("PDSB","PACDRQ",Rep_USR & "\DPRTRQ.txt")
Call BvpEnv("PDSB","PACDKQ",Rep_TMP & "\WKQ.tmp")
Call BvpEnv("PDSB","PACDKD",Rep_TMP & "\WKD.tmp")
Call BvpEnv("PDSB","PACDW1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PDSB","PACDW2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PDSB","PACDW3",Rep_TMP & "\WW3.tmp")
```

```
Call BvpEnv("PDSB","PACDW4",Rep_TMP & "\WW4.tmp")

Call RunCmdLog ("BVPDSB" & CodLang)

If Return < 10  then
Call Msg_Log (Array("1062"))
Return = 0
End if
If Return = 12  then
Call Msg_Log (Array("1063"))
Return = 0
End if
If Return > 12  then
Call Msg_Log (Array("1064"))
End if

Call Err_Cod(Return , 10 , "PDSB" & CodLang)


If BVP_Merge = "YES" then
 Call Msg_Log (Array("1022" , "COPY "))
'-------------------------------------
End If

If BVPACAGP <> " " then
   Call Msg_Log (Array("1022" , BVPACAGP))
'-------------------------------------
Return = WshShell.Run(BVPACAGP , 1 , TRUE )
  Call Err_Cod(Return , 0 , BVPACAGP)
End If



Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 9. DRST - Database restoration

DRST - Introduction

The Database Restoration procedure (DRST) restores files, using the sequential image produced by the Database Backup procedure (DSAV).

Archived transactions can also be retrieved once this procedure has been executed.

Execution conditions

The database must be closed to on-line processing.

Even if the closing of on-line access is not controlled by the procedure, it prevents any other update while the procedure is being executed.

The procedure physically and logically re-initializes the Journal file which must have been saved previously by the DARC procedure.

Abnormal execution

Refer to Subchapter 'Abnormal execution' in Chapter 'The batch procedures'.

Whatever the cause of the abend, the procedure can be restarted as it is, after the problem has been solved.

DEFINITION CONTROL SUB-PROGRAMS

These sub-programs (delivered as COBOL sources) are designed to add specific controls or initializations on the 5 DSMS definitions (Change, Event, Report, Request and Site).

Process:

When the screen is displayed for the first time, there is no access to the control sub-program.

When it is updated, the usual controls are first executed by the Definition screen, and then the sub-program is called. This sub-program will search for fatal errors (F40) and will send a message, whenever relevant, with an update lock, to the calling program which will just display the information.

If no error is found (or after the correction of errors, followed by the usual controls and a new call to the sub-program), the values entered are controlled again and a warning may be sent (F45). The user will then just have to press ENTER to take into account the value previoulsy entered.

Then, via a branching or a new call, the sub-program will be able to assign a new value to some input fields (F50).

Upon the return to the calling program, all the values (entered by the user or assigned by the sub-program) will be controlled again. This screen will then update the Database.

At the beginning, these sources only include 3 examples:

- 1 'WARNING'-type error
- 1 critical error
- 1 initialization.

Their linkage is made up of the displayed fields, the entered fields or some other fields directly or indirectly associated with the definition.

These sub-programs are called via tops indicated in the technical record of the DRST procedure.

There are 10 of them: 5 for on-line processing and 5 for batch processing.

Note: Errors are set using 'PR' (as in VAPAC) ; these fields must be set to 'W' or 'E'.

DRST - Input / Processing / Results

USER INPUT

The following chart lists the DRST procedure's input.

Pos.	Len.	Value	Meaning
2	1	'R'	Line code
3	1	'l'	Language code 'E' or 'F' (optional)
4	1		Journal inhibition flag
		'0'	No inhibition (default option)
		'1'	Inhibition
5	3	'REC'	Restoration and retrieval of archived transactions

Pos.	Len.	Value	Meaning
8	12		12-position table indicating the PFkeys assignment (default: 123456789ABC, but you may move or set to blank one or several values)
20	1		SECURITY SYSTEM INTERFACE
		' '	Retrieval of the previous value or no interface (for creation)
		'&'	Reset to blank = Deactivation
		'R'	RACF
		'S'	TOPSECRET
21	1		USER CONTROL UNDER RACF (ONLINE MODE)
		' '	Retrieval of the previous value
		'&'	Clear = it is possible to enter a user-password different from the one entered at the firt connection
		'N'	It is not possible to enter another user-password
22	1	'C'	Encryption of passwords
		'D'	Decryption of passwords
		' '	Unchanged passwords
			NOTE: You are strongly advised against requesting an encryption or decryption of passwords at the same time as the retrieval of archived transactions (since the action is not performed on the journal).
25	1	'C'	Call of the sub-routine of additional controls for Change definition
		'&'	No call of sub-routine
26	1	'E'	Call of the sub-routine of additional controls for Event definition
		'&'	No call of sub-routine
27	1	'Q'	Call of the sub-routine of additional controls for Query definition
		'&'	No call of sub-routine
28	1	'R'	Call of the sub-routine of additional controls for Report definition
		'&'	No call of sub-routine
29	1	'S'	Call of the sub-routine of additional controls for Site definition
		'&'	No call of sub-routine

OUTPUT REPORT

This procedure prints a report listing the requested options, associated errors, the number of records restored in the database for each file, and the options memorized in the new database.

RESULT

Once this procedure is executed, the current session number is that of the sequential image or that of the most recent transaction, if the retrieval of archived transactions has been requested.

DRST - Description of steps

Validation of journal contents: PDS380

This step is executed only when the Journal file exists. In this case, it checks that the journal has been archived.

Code	Physical name	Type	Label
PACDDJ	Journal dir.: DJ	Input	Journal file
PACDDE	System - Skel. dir: DE	Input	Error message file
PACDRU	User dir.: DRSTRU380.txt	Report	Status of AJ file: It is printed if the journal file has not been archived.

Return codes:

- 0 : The Journal file was archived.
- 4 : The Journal file was not archived.

  (In this case, none of the DRST steps is executed).

Database restoration: PDS400

This step is executed only if the Journal file has been archived.

Code	Physical name	Type	Label
PACDBB	Save dir.: BB	Input	Backup of the files
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDMB		Input	User transactions
PACDDA	DBase dir.: DA	Output	Data file
PACDDC	DBase dir.: DC	Output	VA Pac elements file
PACDDJ	Journal dir.: DJ	Output	Journal file

Code	Physical name	Type	Label
PACDDX	DBase dir.: DX	Output	Cross-reference file
PACDMS	tmp. dir.: MS	Output	Work file (2 records)
PACDRU	User dir.: DRSTRU400.txt	Report	Restoration report

Retrieval of archived journal: PDS450

This step is executed only when there are transactions to be retrieved. It does not cause a 'journalization' of transactions.

Code	Physical name	Type	Label
PACDMS	Tmp. dir.: MS	Input	Work file (2 records)
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDDA	DBase dir.: DA	Input Output	Data file
PACDDC	DBase dir.: DC	Input Output	VA Pac element file
PACDDX	DBase dir.: DX	Input Output	Cross-reference file
PACDBJ	Save dir.: BJ	Input	Archiving of the journal to retrieve
PACDRU	User dir.: DRSTRU450.txt	Report	Update report

DRST - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'    - RELOADING RESTORATION OF THE DSMS DATABASE -
'
' ------------------------------------------------------
'
' INPUT
'  COL 02     : R
'  COL 03     : INITIAL LANGUAGE CODE(F=FRENCH,E=ENGLISH)
'  COL 04     : 1   : INHIBITION OF TRANSACTION LOG
'  COL 05-07  : REC : RETRIEVAL OF ARCHIVED TRANSACTIONS
'  COL 08-19  : (NOT USED)
'  COL 20     : SECURITY SYSTEM (R,S, ,&)
'  COL 21     : USER CONTROL UNDER RACF (N, ,&)
'  COL 22     : CRYPT/UNCRYPT OF PASSWORD (C,D, )
'  COL 23-24  : (NOT USED)
```

```
'  COL 25     : CALL OF SUB-PGM FOR CHANGES (C, ,&)
'  COL 26     : CALL OF SUB-PGM FOR EVENTS (E, ,&)
'  COL 27     : CALL OF SUB-PGM FOR QUERIES (Q, ,&)
'  COL 28     : CALL OF SUB-PGM FOR LAYOUTS (R, ,&)
'  COL 29     : CALL OF SUB-PGM FOR SITES (S, ,&)
'
' IF THE JOURNAL FILE OF TRANSACTIONS ON DISK (DJ) IS NOT
' REINITIALIZED, NO RESTORATION IS EXECUTED.
' IT IS THEREFORE NECESSARY TO EXECUTE THE DARC PROCEDURE
' FIRST.
' ------------------------------------------------------
<job id=DRST>

<script language="VBScript">
Dim MyProc
MyProc = "DRST"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
CodLang = WshShell.RegRead (Rep_D & "\BaseLang")

If  FSO.FileExists(Rep_JOURNAL & "\DJ") Then
Call Msg_Log (Array("1022" , "PDS38" & CodLang))
'-------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
Call BvpEnv("PDS380","PACDRU",Rep_USR & "\DRSTRU380.txt")
Call RunCmdLog ("BVPDS38" & CodLang)
WshVolEnv("RC") = Return
If Return = 4 Then
Call Msg_Log (Array("1058"))
End If
Call Err_Cod(Return , 0 , "PDS38" & CodLang)
End If

Call Msg_Log (Array("1022" , "PDS400"))
'-------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDS400","PACDRU",Rep_USR & "\DRSTRU400.txt")
WshEnv("PACDBB") =  Rep_SAVE & "\BB"
Call BvpEnv("PDS400","PACDMS",Rep_TMP & "\WMS.tmp")
Call RunCmdLog ("BVPDS400")
WshVolEnv("RC") = Return
```

```
Call Err_Cod(Return , 0 , "PDS400")

If  FSO.FileExists( Rep_SAVE & "\BJ"  ) Then
Call Msg_Log (Array("1022" , "PDS450"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDBJ") =  Rep_SAVE & "\BJ"
Call BvpEnv("PDS450","PACDMS",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PDS450","PACDRU",Rep_USR & "\DRSTRU450.txt")
Call RunCmdLog ("BVPDS450")
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PDS450")
End If

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 10. DSAV - Database backup

DSAV - Introduction

The purpose of the backup procedure (DSAV) is to convert the main files that make up DSMS to a BB sequential format.

The backed-up files are :

- The Data file (DA),
- The VA Pac Element file (DC),
- The Cross-reference file (DX).

Execution condition

The database must be closed to on-line processing in order to ensure its consistency during the execution of the DSAV procedure.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

The main cause of an abend is that the database has not been closed to on-line use.

After correction, the procedure can be restarted as it is.

DSAV - Input / Processing / Results

USER INPUT

One optional line code.

Col.	Len.	Value	Designation
2	1	'O'	Line Code
3	3	'ENC'	Encryption of passwords
		'DEC'	Decryption of passwords
		' '	Unchanged passwords

REPORT RESULTS

Once the backup is executed, a report is printed. It includes the number of records saved in each file and the session number.

OUTPUT RESULT

The output is a single sequential file (BB) of variable length, containing the image of the three saved files.

If the database is in an inconsistent state after an abnormal end in the last update, the DSAV procedure is not executed.

Note: The DSAV procedure increments the current session number.

DSAV - Description of steps

Database integrity check: PDSBAS

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir. DE	Input	Error message file
PACDRS	User dir.: DSAVRSBAS.txt	Report	Validity report

Return code:

This utility sends a return code 4 and causes an ABEND if the Database is invalid.

Database backup: PDS500

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input Output	Data file
PACDDC	DBase dir.: DC	Input	VA Pac element file
PACDDX	DBase dir.: DX	Input	Cross-reference file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDMB		Input	User transactions
PACDBB	Save dir.: BB.NEW	Output	Sequential image of files
PACDRU	User dir.: DSAVRU500.txt	Report	Backup report

DSAV - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'            - BACKUP OF THE DSMS DATABASE -
'
' ------------------------------------------------------
'
' INPUT
'  COL 02     : 'O'
'  COL 03-05  : CRYPT, UNCRYPT (ENC,DEC,   )
'
' ------------------------------------------------------
'
<job id=DSAV>

<script language="VBScript">
Dim MyProc
MyProc = "DSAV"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Call Msg_Log (Array("1022" , "PDSBAS"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
Call BvpEnv("PDSBAS","PACDRS",Rep_USR & "\DSAVRSBAS.txt")
Call RunCmdLog ("BVPDSBAS")
WshVolEnv("RC") = Return
If Return = 4 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 0 , "PDSBAS")

Call Msg_Log (Array("1022" , "PDS500"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDMB") = Fic_Input
WshEnv("PACDBB") =  Rep_SAVE & "\BB-new"
Call BvpEnv("PDS500","PACDRU",Rep_USR & "\DSAVRU500.txt")
Call RunCmdLog ("BVPDS500")
Call Err_Cod(Return , 0 , "PDS500")
```

```
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover( Rep_SAVE & "\BB")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'---------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 11. DREO - Reorganization of cross-reference file

DREO - Introduction

The Cross-Reference Reorganization procedure (DREO) rebuilds a sequential image of the database using another sequential image as a starting point. The resulting file is used as input to the Restoration (DRST) procedure.

The operating principle of this procedure is to rebuild the cross-references associated with the data from the 'image' of this data.

Execution conditions

The database can remain open during reorganization since the procedure operates on sequential images of the database (backups).

The updates executed after the constitution of the file back-up used for the reorganization can be retrieved upon the restoration of the reorganized database.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

In case of an abnormal end, the procedure must be restarted from the beginning.

DREO - Input / Processing / Results

USER INPUT

Three different types of user input can be entered, but only one line of each type can be created.

The format of this input is given in the table below.

Pos.	Len.	Value	Meaning
1	1	Not Used	
2	1	'P'	Deletion of Products
	1	'S'	Deletion of Subsidiaries
	1	'X'	Deletion of Products/Subsidiaries

Pos.	Len.	Value	Meaning
3	60	Product code	(20 x 3 char.) if Col.2 = 'P'
	60	Subsid. code	(30 x 2 char.) if Col.2 = 'S'
	60	Prod./ Subsid.	(12 x 5 char.) if Col.2 = 'X'

REPORT

This procedure prints messages stating inconsistencies found in the Data file.

RESULT

The result of this procedure is a reorganized sequential image of the DSMS database, used as input to the Restoration (DRST) procedure.

DREO - Description of steps

Building of indexes (not keywords): PDSR10

Code	Physical name	Type	Label
CARTE		Input	Transactions
PACDBB	Save dir.: BB	Input	DSMS database backup
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDW1	Tmp dir.: W1	Work file	VA Pac elements and data
PACDW2	Tmp dir.: W2	Work file	Keywords and keyword references
PACDW3	Tmp dir.: W3	Work file	Cross-references (not keywords)
PACDRH	User dir.: DREORHR10.txt	Report	Inconsistencies in DSMS data
PACDRK	User dir.: DREORKR10.txt	Report	Reorganization report

Building of keyword indexes: PDSR20

Code	Physical name	Type	Label
PACDW2	Tmp dir.: W2	Work file	Keywords and keyword references
PACDW4	Tmp dir.: W4	Work file	Keywords
PACDW5	Tmp dir.: W5	Work file	Keyword references

Merge of indexes: PDSR30

Code	Physical name	Type	Label
PACDW3	Tmp dir.: W3	Work file	Cross-references (except keywords)
PACDW5	Tmp dir.: W5	Work file	Keyword references
PACDW6	Tmp dir.: W6	Work file	Keyword references

General merge for backup: PDSR40

Code	Physical name	Type	Label
PACDDE	System - Skel. dir. DE	Input	Error message file
PACDW1	Tmp dir.: W1	Work file	VA Pac elements and data
PACDW4	Tmp dir.: W4	Work file	Keywords
PACDW6	Tmp dir.: W6	Work file	Keyword references
PACDBB	Save dir.: BB.NEW	Output	Reorganized DSMS database backup
PACDRR	User dir.: DREORR40.txt	Report	Reorganization report

DREO - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'           - REORGANIZATION OF THE DSMS DATABASE -
'
' ------------------------------------------------------
'
'  OPTIONAL INPUT
'   COL 02     : DELETION OF PRODUCTS, SUBSIDIARIES OR
'                PRODUCT/SUBSIDIARY ENVIRONMENT (P,S,X)
'   COL 03-62  : 20 PRODUCTS, 30 SUBSIDIARIES OR
'                12 PRODUCT/SUBSIDIARY ENVT
'
'
' ------------------------------------------------------
'
<job id=DREO>

<script language="VBScript">
Dim MyProc
MyProc = "DREO"
</script>

<script language="VBScript" src="Incl.vbs"/>
```

```
<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


'Example of Input File extracted :
' Call BvpEnv("PDSR10","PACDMB",RepT_USR & "\xxxx.txt")
' The first line must contain User/Password information
'With  RepT_USR is Global User Directory.


Call Msg_Log (Array("1022" , "PDSR10"))
'-------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("CARTE") = Fic_Input
WshEnv("PACDBB") =  Rep_SAVE & "\BB"
Call BvpEnv("PDSR10","PACDW1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PDSR10","PACDW2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PDSR10","PACDW3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PDSR10","PACDRH",Rep_USR & "\DREORHR10.txt")
Call BvpEnv("PDSR10","PACDRK",Rep_USR & "\DREORKR10.txt")
Call RunCmdLog ("BVPDSR10")
Call Err_Cod(Return , 0 , "PDSR10")

Call Msg_Log (Array("1022" , "PDSR20"))
'-----------------------------------
Call BvpEnv("PDSR20","PACDW2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PDSR20","PACDW4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PDSR20","PACDW5",Rep_TMP & "\WW5.tmp")
Call RunCmdLog ("BVPDSR20")
Call Err_Cod(Return , 0 , "PDSR20")


Call Msg_Log (Array("1022" , "PDSR30"))
'-----------------------------------
Call BvpEnv("PDSR30","PACDW3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PDSR30","PACDW5",Rep_TMP & "\WW5.tmp")
Call BvpEnv("PDSR30","PACDW6",Rep_TMP & "\WW6.tmp")
Call RunCmdLog ("BVPDSR30")
Call Err_Cod(Return , 0 , "PDSR30")

Call Msg_Log (Array("1022" , "PDSR40"))
'-----------------------------------
Call BvpEnv("PDSR40","PACDW1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PDSR40","PACDW4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PDSR40","PACDW6",Rep_TMP & "\WW6.tmp")
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDBB") =  Rep_SAVE & "\BB-new"
Call BvpEnv("PDSR40","PACDRR",Rep_USR & "\DREORRR40.txt")
Call RunCmdLog ("BVPDSR40")
Call Err_Cod(Return , 0 , "PDSR40")


If Return = 0 then
```

```
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\BB" )
End if

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)



Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 12. DEXP - Extraction from VA Pac archived journal

DEXP - Introduction

The Archived Journal Extraction procedure (DEXP) extracts transactions associated with Changes from the VA Pac Archived Journal file, and formats them in order to update, in the DSMS Database, the modified elements corresponding to each Change.

Execution conditions

None.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

If an abnormal end occurs, the procedure can be restarted as it is, after the problem has been solved.

DEXP - Input / Processing / Results

USER INPUT

One '*'-line is required:

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	DSMS user code
11	8	pppppppp	User password

One extraction line is also required:

Pos.	Len.	Value	Meaning
2	1	'J'	Line code (required)
			THE FOLLOWING FIELDS ARE OPTIONAL :
3	1	' '	List of selected transactions
		'N'	No list

Pos.	Len.	Value	Meaning
4	24		Selection in the VA Pac Database:
4	4	nnnn	Session number, begin. of selection
8	4	pppp	Session number, end of selection
			--> Selection on session(s) prohibits selection on date(s)
12	8	CCYYMMDD	Starting date for selection
		'TODAY'	Starting date = current date
20	8	CCYYMMDD	Ending date for selection
		'TODAY'	Ending date = current date (default value if starting date ='today')
28	1		Version of selected transactions
		' '	Selection of all sessions
		'T'	Selection of frozen session
		'Z'	Selection of current session
29	3	ppp	Product code
32	4	xxxx	VA Pac Database logical code
36	3	lll	Code of selected library
39	16		Type of selected entities
55	1	' '	Extraction of transactions made under change 999999
		'N'	No extraction of 999999-change transactions
56	1	' '	Printing of duplicate transactions for the same VA Pac entity
		'N'	No printing of duplicate transactions
57	6	nnnnnn	Change number

REPORT

Extraction report showing the list of formatted transactions.

RESULT

A DSMS database update transaction file to be used as input to the DUPT procedure.

DEXP - Description of steps

Transaction extraction and formatting: PDS600

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PAC7PJ	Tmp dir.: PJ	Input	VA Pac archived journal
PACDMB		Input	User transactions
PACDMV		Output	DUPT update transaction file
PACDRU	User dir.: DEXPRU600.txt	Report	Report on selection request

Return codes:

- 0 : No error and no list requested
- 4 : No error and printout of the transactions list
- 8 : Error on the user line code or parameter line
- 12 : I/O error on a file

Printing of DSMS update transactions: PDS610

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	Systaem- Skel. dir.: DE	Input	Error message file
PACDMV		Input	DSMS update transactions file
PACDRU	User dir.: DEXPRU610.txt	Report	List of update transactions

Return codes:

- 0 : No error
- 12 : I/O error on a file

DEXP - Execution script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' -------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
'           - EXTRACTION OF PACBASE JOURNAL     -
' -------------------------------------------------------
'
```

```
'  .. A DSMS USER AND PASSWORD LINE
' COL 02     : *
' COL 03     : DSMS USER CODE
' COL 11     : PASSWORD
'  .. COMMAND LINE(S) FOR EXTRACTION
' COL 02     : J
' COL 03     : ' ' SELECTED TRANSACTIONS LIST
'             : 'N' NO LIST OF SELECTED TRANSACTIONS
' COL 04-07  : STARTING SESSION NUMBER
' COL 08-11  : ENDING SESSION NUMBER
' COL 12-19  : STARTING DATE (CCYYMMDD)
' COL 20-27  : ENDING DATE (CCYYMMDD)
' COL 28     : VERSION OF SELECTED TRANSACTIONS
'             : ' ' ALL SESSIONS
' COL 36-38  : LIBRARY CODE
' COL 39-54  : TYPE OF ENTITIES TO BE SELECTED
' COL 55     : EXTRACT OF TRANSAC. FOR CHANGE 999999
' COL 56     : PRINTING OF ALL TRANSACTIONS
' COL 57-62  : CHANGE NUMBER
'
' ------------------------------------------------------
'
<job id=DEXP>

<script language="VBScript">
Dim MyProc
MyProc = "DEXP"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Call Msg_Log (Array("1022" , "PDS600"))
'-----------------------------------

CALL BVPENV("PDS600","PAC7AN",REP_BASEP & "\AN")
CALL BVPENV("PDS600","PAC7AR",REP_BASEP & "\AR")

WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDS600","PACDMV",Rep_USR & "\MVEXP.txt")
Call BvpEnv("PDS600","PACDRU",Rep_USR & "\DEXP600RU.txt")
Call BvpEnv("PDS600","PAC7PJ",Rep_SAVE & "\PJ")
Call RunCmdLog ("BVPDS600")


If Return = 0  then
Call Msg_Log (Array("1045"))
Return = 0
End if
```

```
If Return = 4 then
Call Msg_Log (Array("1046"))
Return = 0
End if

If Return = 8  then
Call Msg_Log (Array("1030"))
End if
If Return = 12  then
Call Msg_Log (Array("1027"))
End if
Call Err_Cod(Return , 4 , "PDS600")


Call Msg_Log (Array("1022" , "PDS610"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
Call BvpEnv("PDS610","PACDMV",Rep_USR & "\MVEXP.txt")
Call BvpEnv("PDS610","PACDRU",Rep_USR & "\DEXP610RU.txt")
Call RunCmdLog ("BVPDS610")

Call Err_Cod(Return , 0 , "PDS610")



Call Msg_Log (Array("1024"))
'---------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'---------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 13. DEXT - Extraction of entities

DEXT - Introduction

The Entity Extraction procedure (DEXT) extracts all DSMS entities and formats them into batch transactions to be used as input to the DSMS Database Update procedure (DUPT).

Principle

In order to select the extraction of Changes, Events or Sites, the procedure uses Queries (″Q″ entities) that must have been previously defined in the DSMS Database. These three types of extraction must be requested in the above order.

The Query code should also be specified in the extraction request (see 'User Input').

The screen Report (″R″ entity) associated with the Query used for the extraction does not interfere in the extraction.

Execution conditions

None.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

If an abnormal end occurs, the procedure can be restarted as it is after the problem has been solved.

DEXT - Input / Processing / Results

USER INPUT

One '*'-line is required:

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	DSMS User code
11	8	pppppppp	User password

Pos.	Len.	Value	Meaning
19	3	ppp	Product code
22	2	su	Subsidiary code
24	1	l	Language code

Four types of extractions are available. One line per request is necessary:

Pos.	Len.	Value	Meaning
02	03	'PL'	Locking of databases
02	03	Txx	Codes of the Txx table (all tables except TRA)

- Queries / Reports:

Pos.	Len.	Value	Meaning
02	04	X QC	Query on Changes
		X QE	Query on Events
		X QS	Query on Sites
02	04	X RC	Report on Changes
		X RE	Report on Events
		X RS	Report on Sites
12	08	uuuuuuuu	Owner of the Query or Report (Default=logged-in user)

- Lists

Pos.	Len.	Value	Meaning
02	04	LCQC	Queries on Changes
		LCQE	Queries on Events
		LCQS	Queries on Sites
02	04	LCRC	Reports on Changes
		LCRE	Reports on Events
		LCRS	Reports on Sites
12	08	uuuuuuuu	Owner of Queries or Reports

- Keywords

Pos.	Len.	Value	Meaning
02	04	LAKC	Stand-alone keywords of Changes
		LGKC	All Keywords of Changes

Pos.	Len.	Value	Meaning
06	01	l	Language code of Keywords (Default=Language of logged-in user)
02	04	LAKE	Native stand-alone Keywords of Events
		LGKE	All native Keywords of Events
02	04	LAKT	Techn. stand-alone Keywords of Events
		LGKT	All techn. Keywords of Events

Extraction via user request (99 requests maximum)

Pos.	Len.	Value	Meaning
2	1	'Q'	
3	1	'C'	For a query on Changes
		'E'	For a query on Events
		'S'	For a query on Sites
5	6	rrrrrr	User Query code (required) - 'Q' Entity use
5	6	mmmmmm	Report code (optional)
17	1	d	Delimiter (optional)
			Parameter settings:
18	1	s	Symbol (optional)
19	1	x	Separator (optional)
20	54	.....	Parameter values (optional)
			If some optional fields were not completed, default values will be used. They come from the User Query's definition lines found in the Database.

PRINTED OUTPUT

Extraction report showing the number of extracted transactions.

RESULT

DSMS database update transactions to be used as input to the DUPT procedure.

This procedure displays a general return code:

Code	Description
0	OK
8	Error on the user line code (*) or on a parameter line
12	I/O error or inconsistent DSMS database
16	Sort error

DEXT - Description of steps

This procedure calls a single program (PDSEX) that acts as a flow monitor for all programs, which are then considered as its sub-programs.

The procedure includes the following steps:

Extractions: PDSEX

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDC	DBase dir.: DC	Input	VA Pac element file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDMB		Input	Extraction requests
PACDKQ	Tmp dir.: WKQ	Work file	Queries
PACDIM		Output	Extracted batch transactions
PACDIA	User dir.: DEXTIAEX.txt	Report	Flow report
PACDRU	User dir.: DEXTRUEX.txt	Report	Extraction request report
PACDW0	Tmp dir.: WK0	Work file	Temporary file
PACDW1	Tmp dir.: WW1	Work file	Temporary file
PACDW2	Tmp dir.: WW2	Work file	Temporary file
PACDW3	Tmp dir.: WW3	Work file	Temporary file
PACDW4	Tmp dir.: WW4	Work file	Temporary file
PACDW5	Tmp dir.: WW5	Work file	Temporary file
PACDWI	tmp dir.: WWI	Work file	Temporary file

DEXT - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
'           - EXTRACTION OF PACBASE JOURNAL     -
' ------------------------------------------------------
'
' INPUT :
' .. IDENTIFICATION LINE
'  COL 02     : *
'  COL 03     : DSMS USER CODE
'  COL 11     : PASSWORD
'  COL 19-21  : PRODUCT CODE
'  COL 22-23  : SUBSIDIARY CODE
'  COL 24     : LANGUAGE CODE
'
'  ..  EXTRACT COMMAND LINE(S)
'      ----------------------
'  COL 02-05  : TYPE OF EXTRACTION
'  -- EXTRACTION BY USER QUERY :
'  COL 05-10  : QUERY CODE
'  COL 17     : DELIMITER                   <--- OPTIONAL
'  COL 18     : SYMBOL                      <--- OPTIONAL
'  COL 19     : SEPARATOR                   <--- OPTIONAL
'  COL 20-73  : PARAMETERS VALUES           <--- OPTIONAL
' --- EXTRACTION OF QUERIES/LAYOUT :
' -COL 06-11  : QUERY OR LAYOUT CODE
'  COL 12-19  : OWNER OF THE QUERY/LAYOUT   <--- OPTIONAL
'
' ------------------------------------------------------
'
<job id=DEXT>

<script language="VBScript">
Dim MyProc
MyProc = "DEXT"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
CodLang = WshShell.RegRead (Rep_D & "\BaseLang")

 Call Msg_Log (Array("1022" , "PDSEX" & CodLang))

'-------------------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
```

```
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDSEX","PACDIM",Rep_USR & "\MVEXT.txt")
Call BvpEnv("PDSEX","PACDRU",Rep_USR & "\DEXTRU.txt")
Call BvpEnv("PDSEX","PACDIA",Rep_USR & "\DEXTIA.txt")
Call BvpEnv("PDSEX","PACDKQ",Rep_TMP & "\WKQ.tmp")
Call BvpEnv("PDSEX","PACDW0",Rep_TMP & "\WK0.tmp")
Call BvpEnv("PDSEX","PACDW1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PDSEX","PACDW2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PDSEX","PACDW3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PDSEX","PACDW4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PDSEX","PACDW5",Rep_TMP & "\WW5.tmp")
Call BvpEnv("PDSEX","PACDWI",Rep_TMP & "\WWI.tmp")

Call RunCmdLog ("BVPDSEX" & CodLang)

If Return = 8 then
Call Msg_Log (Array("1030"))
Return = 0
End if

If Return = 12  then
Call Msg_Log (Array("1032"))
Return = 0
End if

If Return = 16  then
Call Msg_Log (Array("1064"))
End if

Call Err_Cod(Return , 10 , "PDSEX" & CodLang)

Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 14. DUPT - Batch update of entities

DUPT - Introduction

The Batch Update of Entities procedure (DUPT) updates the DSMS entities with transactions from the DEXT, DEXP and/or DXBJ procedures.

Transactions can also be entered directly in a file, using an editor. For a complete description of the batch transactions, see the 'Batch Transactions structure', in the appendix of the DSMS Reference Manual.

Execution condition

The DSMS files must be closed to on-line use.

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'. If an abend occurs, and after the problem is solved,

- if a batch rollback can be executed, you can restart the procedure as it is,
- otherwise, you must first execute a restoration with the retrieval of archived transactions.

CAUTION:

This procedure performs a GLOBAL update. Therefore, make sure that all the data fields have been filled in. The data fields that are not filled in will automatically be reset to blank.

The Change, Event and Site definition screens require two update lines, and both lines must be filled.

DSMS automatically allocates numbers to Events or Changes when they are created. However, for its creation, an Event or Change must be allocated a temporary number. For example, to create a Change: C000001, where 000001 is the temporary number that DSMS will automatically replace with a unique number.

You must set the action code to 'C', since the system does not provide for implicit creation.

Several Changes or Events can be created simultaneously. In this case, each Change or Event being created must be allocated a different temporary number. For example, to create 3 Changes simultaneously: C000001, C000002 and C000003.

Note: Each transaction stream can only contain 2,520 changes and 2,520 events maximum (internal limit of the program).

DUPT - Input / Processing / Results

USER INPUT

- One Parameter line (optional).
- One Identification line per Product/Subsidiary concerned by the updates (required).
- Update transactions extracted and formatted by the DEXT, DEXP or DXBJ procedures.
- The user must add at least one identification line in front of update transactions.

Parameter line (optional)

Col	Len	Value	Description
2	1	$	LINE CODE
3	1		UPDATE MODE / SORT ORDER
			Defines the update or processing mode to be used by ALL userids for this execution of the DSMS batch procedure.
		A	NORMAL UPDATE MODE
			- Transactions sorted in ascending order before any update is applied (i.e. entity definitions are processed before sub-screens.)
			- Update mode specified for each sign-on record.
		D	DELETE MODE
			- Transactions sorted in descending order before any update is applied.
			- All transactions processed as Deletions - Action Code D'.
			- Sign-on records must specify 'NORMAL' mode - all other modes are considered as errors.
4	1		REPORT FORMAT INDICATOR
		1	SINGLE REPORT FORMAT

Col	Len	Value	Description
			- One 'END OF REPORT' line is produced.
			- The transaction 'INPUT NUMBER' is simply incremented by one for each transaction.
		2	SIGN-ON / USERID FORMAT 1
			- An 'END OF REPORT' line is produced for each userid / sign-on record.
			- The transaction 'INPUT NUMBER' is reset to one for each sign-on record. The sign-on record will appear as transaction number one.
		3	SIGN-ON / USERID FORMAT 2
			- An 'END OF REPORT' line is produced for each userid / sign-on record.
			- The transaction 'INPUT NUMBER' is reset to zero for each sign-on record. The sign-on record will appear as transaction number zero.

If the parameter line is not entered, ' $A1' is assumed.

Sign-on line format (required)

Col	Len	Value	Description
1	1		ACTION CODE / UPDATE MODE
			This field defines the update mode processing to be used for this userid.
		blank	NORMAL UPDATE MODE.
			- Works like DSMS in on-line mode.
			- If an Event or Change is created, all following sub-screen transactions will be modified accordingly.
		V	VERSION CONTROL MODE.
			- All batch transactions will be processed with Action Code 'C' (create).
			- The external reference fields on Event and Change Definitions will be filled in.
			- The associated change fields on Event Definitions will be converted to the 'new' Change Number - the number assigned when the Change is created.
		R	REORGANIZATION MODE.

Col	Len	Value	Description
			The same as 'V' except that the external reference fields' content will not be altered.
2	1	*	SIGN-ON RECORD CODE
3	8	...	DSMS USER
11	8	...	DSMS USER PASSWORD
19	3	ppp	PRODUCT CODE to which updates apply.
22	2	ss	SUBSIDIARY CODE to which batch updates apply.
24	1	blank	Unused
25	9		EXTERNAL REFERENCE VALUES
			The value of the next three fields is used to create Event and Change external references if the update mode is 'V'.
25	4	dddA	- DSMS external Database code
29	3	ppp	- DSMS external Product code
32	2	ss	- DSMS external Subsidiary code
34	1		BLANK LINE AFTER ERROR INDICATOR
		blank	A blank line is printed after each error message on the report.
		N	Blank lines are not printed after error messages on the report.
35	1		REPORT PAGE BREAK INDICATOR
		blank	Page break only when the number of lines per page exceeds the maximum number
		T	Page break for each new type of transaction
		E	Page break for each transaction type of each entity
36	1		TRANSACTION SORT INDICATOR
		blank	The transactions are sorted by type before they are processed.
		N	The transactions are processed in their order of arrival.

REPORT

The printout generated by this procedure is an update report, with comments about irregularities or inconsistencies encountered during execution.

RESULT

The result of this procedure is:

- A DSMS database ready for on-line or batch processing,
- A Journal file of the transactions which have modified the database, if 'journalization' was not inhibited during the last restoration.

Note: This procedure increments the session number if it is the first access to the database for the current day.

DUPT - Description of steps

Database integrity check: PDSBAS

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir. DE	Input	Error message file
PACDRS	User dir.: DSAVRSBAS.txt	Report	Validity report

Return code:

This utility sends a return code 4 and causes an ABEND if the Database is invalid.

Update of the DSMS database: PDSUP0

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input Output	Data file
PACDDC	DBase dir.: DC	Input Output	VA Pac element file
PACDDX	DBase dir.: DX	Input Output	Cross-reference file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDIM		Input	Update transactions obtained via the DEXP procedure
PACDDJ	Journal dir.: DJ	Output	Journal file
PACDRP	User dir.: DUPTRPUP0.txt	Report	Update review

Return codes

- 0: No error

- 8: Error on the user line or the parameter input line
- 12: I/O error on a file

DUPT - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'          - UPDATE OF THE DSMS DATABASE -
'
' ------------------------------------------------------
'
' INPUT :
' .. PARAMETERS LINE (OPTIONAL)
'  COL 02     : $
'  COL 03     : UPDATE MODE (A,D)
'  COL 04     : REPORT FORMAT INDICATOR (1,2,3)
' .. IDENTIFICATTION LINE (MANDATORY)
'  COL 01     : ACTION CODE / UPDATE MODE (V,R, )
'  COL 02     : *
'  COL 03-10  : USER CODE
'  COL 11-18  : PASSWORD
'  COL 19-21  : PRODUCT CODE
'  COL 22-23  : SUBSIDIARY CODE
'  COL 24     : (NOT USED)
'  COL 25-31  : EXTERNAL REFERENCE VALUE (DATABASE,
'               PRODUCT, SUBSIDIARY)
'  COL 34     : BLANK LINE AFTER ERROR ( ,N)
'  COL 35     : REPORT PAGE BREAK INDICATOR ( ,T,E)
'  COL 36     : TRANSACTION SORT INDICATOR ( ,N)
'
'  .. COMMAND LINES
'
' ------------------------------------------------------
'
<job id=DUPT>

<script language="VBScript">
Dim MyProc
MyProc = "DUPT"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Call Msg_Log (Array("1022" , "PDSBAS"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
```

```
Call BvpEnv("PDSBAS","PACDRS",Rep_USR & "\DUPTRSBAS.txt")
Call RunCmdLog ("BVPDSBAS")
WshVolEnv("RC") = Return
If Return = 4 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 0 , "PDSBAS")


Call Msg_Log (Array("1022" , "PDSUP0"))
'-----------------------------------
WshEnv("PACDIM") = Fic_Input

'Example of Input File extracted  :
' Call BvpEnv("PDSUP0","PACDxx",RepT_USR & "\XXxx.txt")
'With  RepT_USR is Global User Directory.

WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDDJ") = Rep_JOURNAL & "\DJ"
Call BvpEnv("PDSUP0","PACDRP",Rep_USR & "\DUPTRPUP0.txt")
Call RunCmdLog ("BVPDSUP0")

Call Err_Cod(Return , 0 , "PDSUP0")


Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 15. DINI - File initialization

DINI - Introduction

The DINI procedure initializes the files needed for the installation of a new DSMS database.

It provides an initial backup of the DSMS files, which must be loaded by the Database Restoration (DRST) procedure.

Execution conditions

None.

However, the parameters of the new DSMS database must have been previously defined, and must be different from the parameters in any other existing DSMS database on the site.

The initial allocation and loading of DSMS components must have been executed (see the Installation Process).

Abnormal execution

Refer to Subchapter 'Abnormal Execution' in Chapter 'The batch procedures'.

Whatever the cause of the abend, the procedure can be restarted as it is after the problem has been solved.

DINI - Input / Processing / Results

USER INPUT

The structure of the input is as follows:

Pos.	Len.	Value	Meaning
2	1	'I'	Line code
3	1	'l'	Initial language code (E by default: English)
4	1		This field is ONLY used with DOS/VSE
		'I'	Default option for all hardware
		'N'	If CURRENT-DATE = DD/MM/YY in DOS/VSE

REPORT

This procedure prints a report which lists the memorized options and the number of initial records of the DSMS database files.

RESULT

The result is an initial backup which includes:

- an initial user, whose userid is '********' and whose password is '********' (See the paragraph that follows: INITIAL CONNECTION.)
- a record in the Language Table corresponding to the language code indicated in the user input.

IMPORTANT:

INITIAL CONNECTION:

The Database Restoration (DRST) procedure must be executed after the DINI procedure. After a successful execution of the DRST procedure, the DSMS database is installed.

Check that the on-line access to the new DSMS database is operational.

The initial connection to the DSMS database is executed as follows:

- Access the DSMS database.
- On the Sign-on screen, enter '********' as the user code and '********' as the password, then press the ENTER key.
- Among the choices listed on the menu, only those marked with a '*' may be accessed. They correspond to the Tables which must be updated for a proper operation of DSMS. The information must be entered in the Tables in the following order:
  - In the Languages Table (CHOICE: 'TLA'): the codes and labels of the languages used.
  - In the Products Table (CHOICE: 'TPR'): the product codes and labels.
  - In the Subsidiaries Table (CHOICE: 'TSU'): the subsidiary codes and labels.
  - In the User Parameters Tables (CHOICES: 'TUD', 'TUG', 'TUP' and 'TUS'): user codes and authorizations.

(For more details on the management of these tables, see the DSMS Reference Manual).

The '********' user code cannot be deleted; after the User Parameters Tables are updated, the DSMS Database Manager can change its password in order to prevent other users from using this code.

DINI - Description of steps

Initial database backup: PDSINI

Code	Physical name	Type	Label
PACDMB		Input	Initialization transaction
PACDDE	System - Skel. dir.: DE	Input	Error messages
PACDBB	Save dir.: BB.NEW	Output	Sequential image of files
PACDRU	User dir.: DINIRUINI.txt	Report	Backup report

DINI - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'             - INITIALIZATION OF THE DSMS DATABASE -
'
' ------------------------------------------------------
'
' INPUT
' COL  2     : I
' COL  3     : INITIAL LANGUAGE CODE
'              ( F=FRENCH, E=ENGLISH)
' COL  4     : MACHINE DATE FORMAT (I FOR MM/DD/YY)
'            :                     (N FOR DD/MM/YY)
'
' ------------------------------------------------------
'
<job id=DINI>

<script language="VBScript">
Dim MyProc
MyProc = "DINI"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Call Msg_Log (Array("1022" , "PDSINI"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
```

```
Call BvpEnv("PDSINI","PACDRU",Rep_USR & "\DINIRUINI.txt")
WshEnv("PACDBB") =  Rep_SAVE & "\BB-new"
WshEnv("PACDMB") = Fic_Input
Call RunCmdLog ("BVPDSINI")
Call Err_Cod(Return , 0 , "PDSINI")


Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)


If Return = 0 Then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\BB")
End If


Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 16. DXBJ - Journal extraction for update

DXBJ - Introduction

The DXBJ procedure extracts, from the DSMS journal file, all the transactions which correspond to a date/time interval or to a given user, and transforms them into update transactions.

Execution conditions

None.

Abnormal execution

Refer to Chapter 'The batch procedures', Subchapter 'Abnormal execution'.

Whatever the cause of the abend, the procedure can be restarted as it is once the problem has been solved.

DXBJ - Input / Processing / Results

USER INPUT

One '*'-line is required:

Pos.	Len.	Value	Meaning
2	1	'*'	line code
3	8	uuuuuuuu	DSMS User code
11	8	pppppppp	User password
			OPTIONAL:
19	3	ppp	Product code
22	2	su	Subsidiary code
24	1	'F' or 'E'	Language code
			USERS/PASSWORDS IN OUTPUT TRANSAC.

One line per extraction request:

Pos.	Len.	Value	Meaning
2	1	'K'	Line code

Pos.	Len.	Value	Meaning
3	1	' '	List of selected transactions
		'N'	No list
4	8	CCYYMMDD	Starting date for selection
12	8	CCYYMMDD	Ending date for selection
20	6	HHMMSS	Starting time for selection
26	6	HHMMSS	Ending time for selection
32	8	uuuuuuuu	Selected user code
40	1	' '	User codes present in journal file without password.
		'T'	User codes present in journal file with passwords if sufficient authorization.
		'1'	User code and password, detailed in next columns.
41	8	uuuuuuuu	User code for output transactions (if column 40 = 1)
48	8	mmmmmmmm	Password for output transactions (if column 40 = 1)

REPORT

Extraction report and, upon request, the list of formatted transactions.

RESULT

A DSMS update transactions file to be used as input to the DUPT procedure. An 'N' is entered in column 36 of the user line for DUPT not to sort these transactions.

DXBJ - Description of steps

Extraction and formatting of transactions: PDS700

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir.: DE	Input	Error message file
PACDBJ	Save dir.: BJ	Input	Archived DSMS journal
PACDMB		Input	User transactions
PACDIM	User dir.: MVDXBJ	Output	Update transaction file for DUPT
PACDRK	User dir.: DXBJRK700.txt	Report	Extraction review
PACDSK	User dir.: DXBJSK700.txt	Report	Transaction printout

Return codes:

- 0: No error
- 8: Error on the user '*' line or parameter line.
  The environment definition is missing.
- 12: File access error.
  The technical record is missing.

DXBJ - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
'           - EXTRACTION OF DSMS JOURNAL      -
' ------------------------------------------------------
'
'  .. A DSMS USER AND PASSWORD LINE
' COL 02     : *
' COL 03     : DSMS USER CODE
' COL 11     : PASSWORD
' COL 19     : PRODUCT CODE       (OPTIONAL)
' COL 22     : SUBSIDIARY CODE    (OPTIONAL)
' COL 24     : LANGUAGE           (OPTIONAL)
'  .. COMMAND LINE(S) FOR EXTRACTION
' COL 02     : K
' COL 03     : ' ' SELECTED TRANSACTIONS LIST
'            : 'N' NO LIST OF SELECTED TRANSACTIONS
' COL 04-11  : STARTING DATE (CCYYMMDD)
' COL 12-19  : ENDING   DATE (CCYYMMDD)
' COL 20-25  : STARTING HOUR (HHMMSS)
' COL 26-31  : ENDING   HOUR (HHMMSS)
' COL 32-39  : USER CODE
'
' ------------------------------------------------------
'
<job id=DXBJ>

<script language="VBScript">
Dim MyProc
MyProc = "DXBJ"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


'Example of Output File reuse in next procedure :
' Call BvpEnv("PROC","PACxxx",RepT_USR & "\XXX.txt")
```

```
'                        With RepT_USR is Global User Directory.


Call Msg_Log (Array("1022" , "PDS700"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") = Rep_BASE & "\DA"
WshEnv("PACDMB") = Fic_Input
Call BvpEnv("PDS700","PACDIM",RepT_USR & "\MVDXBJ")
Call BvpEnv("PDS700","PACDBJ",Rep_SAVE & "\BJ")
WshEnv("PACDBJ") = Rep_SAVE & "\BJ"
Call BvpEnv("PDS700","PACDRK",Rep_USR & "\DXBJ700RK.txt")
Call BvpEnv("PDS700","PACDSK",Rep_USR & "\DXBJ700SK.txt")
Call RunCmdLog ("BVPDS700")
Call Err_Cod(Return , 0 , "PDS700")

Call Msg_Log (Array("1024"))
'---------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'---------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 17. DREN - Code and keyword update

DREN - Introduction

The Code and Keyword Update procedure (DREN) is used to define new codes (table or site) or new keywords to replace those defined and used until then in the tables, thesaurus, and entities.

Execution condition

This procedure works from a sequential backup and/or an archived journal, and must therefore be preceded by a backup and/or an archiving.

Abnormal execution

See Subchapter 'Abnormal Execution', in Chapter 'The batch procedures'.

Whatever the cause of the abend, the procedure can be restarted as it is once the problem has been solved.

DREN - Input / Processing / Results

USER INPUT

One '*' line (required):

Col.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	DSMS User Code
11	8	pppppppp	Password
			OPTIONAL
19	3	ppp	Modifications of the entities which depend on the product code 'ppp'
		'***'	Modifications of the entities which depend on all the product codes
22	2	ss	Modifications of the entities which depend on the subsidiary code 'ss'
		'**'	Modifications of the entities which depend on all the subsidiary codes
24	1	'E' or 'F'	Language code

Col.	Len.	Value	Meaning
			REQUIRED: AT LEAST ONE OF THESE AREAS SET TO '1'
25	1	' '	No modification of the backup
		'1'	Modifications of the backup
26	1	' '	No modification of the archiving
		'1'	Modifications of the archiving

Command lines (500 maxi)

Col.	Len.	Value	Meaning
2	3	'Txx'	table choice (same as on-line)
		'Kxx'	keyword choice (with xx = 'T ' for technical keywords, xx = 'E ' for native keywords and xx = 'Cl' for Change keywords (language l))
		'S '	site choice
5	9		old site code
14	1		not used
15	3		old site sub-code
18	9		new site code
27	1		not used
28	3		new site sub-code

Notes:

- The codes (old and new) must be preceded by 'C', 'E' or 'S' for the TST table, by 'C' or 'E' for the TGR and TTY tables, and by 'F' or 'R' for the TAT table.
- It is not possible to invert two codes (for example, change 'AA' to 'BB', and 'BB' to 'AA'). However, it is possible to rename a code (with an unknown one), and to reuse the old code to transform other codes (for example: 'AA' becomes 'BB' while 'CC' and 'DD' become 'AA'; in this case the command AA/BB must be written before CC/AA and DD/AA).
- The new codes assigned to products, subsidiaries or sites must not already exist (in the same subsidiary for a site).
- The two parts of the site code (9 and 3 characters) cannot be modified separately.
- For the TVE table, you can request the following updates:
  - Technical package alone

- Technical package and release
- Technical package, release and hardware
- Technical package, release, hardware and version (with or without language code)
- Release alone
- Hardware alone
- Version number (with or without language code)

Isolated parts should be aligned as if the other parts were there.

Ascending consistency checks are performed. The changes requested on the preceding lines must be taken into account.

- The label associated with the new code can be either that of the old code or that of the new code if it already existed. This choice is made while the file is sorted and is therefore unpredictable.
- For tables depending on a product (TOP, TPH and TVE), the product's code must be clearly specified on the '*' line.

PRINTED OUTPUT

Report on changes concerning the backup and/or the archiving.

Note on counters:

They count the total number of updates but not the number of modified records (there can be several modifications on the same record).

RESULT

If the change was made on the archive (1 in column 26), a new version of the Journal's sequential backup is produced.

If the change was made on the Database backup (1 in column 25), the result is a new version of the Database sequential backup which should be reorganized via the DREO procedure before being restored.

Return code

Code	**Meaning**
0	OK
8	Error on the '*' line or on a command line
10	Invalid absence of save/archiving flag

Code	Meaning
11	Erroneous character in the save/archiving flags areas (Possible values: ″ ″,″0″, ″1″.)
12	I-O error or inconsistent DSMS database
16	Sort error

DREN - Description of steps

This procedure calls a single program (PDSMS) which is used as a flow monitor for various programs considered as sub-routines of this monitor. It includes the following steps:

Updates: PDSMS

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDC	DBase dir.: DC	Input	VA Pac element file
PACDDE	System - Skel. dir.: DE	Input	Error messages
PACDDX	DBase dir.: DX	Input	Cross-references
PACDBB	Save dir.: BB	Input	DSMS backup
PACDBJ	Save dir.: BJ	Input	DSMS archiving
PACDMB		Input	User queries
PACDW0	Tmp dir.: W0	Work file	Update requests
PACDW1	Tmp dir.: W1	Work file	Partial backup (sorted)
PACDW2	Tmp dir.: W2	Work file	Partial backup (not sorted)
PACDB3	Save dir.: BB.NEW	Output	Modified backup
PACDJB	Save dir.: BJ.NEW	Output	Modified archive
PACDIA	User dir.: DRENIAMS.txt	Report	Flow report
PACDIK	User dir.: DRENIKMS.txt	Report	List of commands on the backup
PACDJK	User dir.: DRENJKMS.txt	Report	Update report (backup)
PACDIS	User dir.: DRENISMS.txt	Report	Merging report (backup)
PACDKK	User dir.: DRENKKMS.txt	Report	List of commands on archiving
PACDLK	User dir.: DRENLKMS.txt	Report	Update report (archive)

DREN - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'      - CHANGE OF TABLE AND SITE CODES, AND KEYWORDS
'
' ------------------------------------------------------
'
' INPUT :
' .. IDENTIFICATION LINE
'  COL 02     : *
'  COL 03     : DSMS USER CODE
'  COL 11     : PASSWORD
'  COL 19-21  : PRODUCT CODE OR '***'
'  COL 22-23  : SUBSIDIARY CODE OR '**'
'  COL 24     : LANGUAGE CODE
'  COL 25     : MODIFICATIONS ON SAVE (1, )
'  COL 26     : MODIFICATIONS ON ARCHIVE (1, )
' ..  MODIFICATION(S) COMMAND LINE(S)
'  COL 02-04  : TYPE OF MODIFICATION
'  COL 05-17  : OLD CODE
'  COL 18-30  : NEW CODE
'
' ------------------------------------------------------
'
<job id=DREN>

<script language="VBScript">
Dim MyProc
MyProc = "DREN"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
CodLang = WshShell.RegRead (Rep_D & "\BaseLang")

 Call Msg_Log (Array("1022" , "PDSMS" & CodLang))

'-------------------------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDMB") = Fic_Input
WshEnv("PACDBB") = Rep_SAVE & "\BB"
WshEnv("PACDB3") = Rep_SAVE & "\BB-new"
WshEnv("PACDBJ") = Rep_SAVE & "\BJ"
```

```
WshEnv("PACDJB") = Rep_SAVE & "\BJ-new"
Call BvpEnv("PDSMS","PACDW0",Rep_TMP & "\WW0.tmp")
Call BvpEnv("PDSMS","PACDW1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PDSMS","PACDW2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PDSMS","PACDIA",Rep_USR & "\DRENIAMS.txt")
Call BvpEnv("PDSMS","PACDIK",Rep_USR & "\DRENIKMS.txt")
Call BvpEnv("PDSMS","PACDJK",Rep_USR & "\DRENJKMS.txt")
Call BvpEnv("PDSMS","PACDIS",Rep_USR & "\DRENISMS.txt")
Call BvpEnv("PDSMS","PACDKK",Rep_USR & "\DRENKKMS.txt")
Call BvpEnv("PDSMS","PACDLK",Rep_USR & "\DRENLKMS.txt")

Call RunCmdLog ("BVPDSMS" & CodLang)

If Return = 8 Then
Call Msg_Log (Array("1030"))
End If

Call Err_Cod(Return , 10 , "PDSMS" & CodLang)

If Return = 0 Then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\BB")
Call Turnover(Rep_SAVE & "\BJ")
End If

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 18. DPDF - Generated programs DAF pre-processor

DPDF - Introduction

The DPDF procedure processes user generated programs that contain SQL requests for Database access through DAF operators.

Execution condition

None.

Implementation

The DPDF procedure may be executed in several ways:

- Either after a program generation via GPRT, whose generated output is used as input to DPDF, before being passed on for compilation or storing in a source-program library.
- Or by calling the procedure in the optional before/after control cards of the program. In this case, the correct JCL must have been entered in the selected options, which are updated in Administrator Workench, in the 'Optional Command Lines Sets' tab.

DPDF - Input / Processing / Results

USER INPUT

It is the COBOL source of the programs containing DAF operators which must be solved by the pre-processor before being compiled.

Each program contains, after the IDENTIFICATION DIVISION line, a command line for the pre-processor:

Pos.	Len.	Value	Meaning
1	6	nnnnnn	COBOL line number
7	1	'*'	Comments
8	5	'TP '	On-line program, or
		'BATCH'	Batch program
13	6	'LIB:'	Fixed label
19	3	bbb	Library code

Pos.	Len.	Value	Meaning
22	1	blank	Not used
23	5	nnnns	Session number - Session status
28	1	blank	Not used
29	2	--	Generation variant(s)
31	5	'AR:'	Fixed label
36	1	l	Database language code
37	5	'SC:'	Batch language program skeleton
		'SG:'	OLSD program skeleton
		'SR:'	COBOL Generator program skeleton
42	1	l	Skeleton language
43	1	blank	Not used
44	6	'SINGLE'	Single quotes, or
		'DOUBLE'	Double quotes

Examples:

```
000020*TP    LIB: APP 2345  00 AR: F SG: F SINGLE
000020*BATCH LIB: APP 2300T 4  AR: F SC: F DOUBLE
```

This line is automatically generated by the GPRT procedure.

PRINTED OUTPUT

The procedure prints the list of errors, if any.

RESULT

The result of the execution is a COBOL source file in which all DAF operators have been solved, and all the calls to Database batch or on-line access routines have been generated.

DPDF - Description of steps

The DPDF procedure calls a single program which acts as a flow monitor for various programs, considered as sub-programs of this monitor. It includes the following step:

Generated program's pre-processor: DAFD10

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir.: DE	Input	Error message file
DAF80	User dir.: DAF	Input	Generated programs
COB80	User dir.: COB	Output	Generated programs to be compiled
DAFREP	User dir.: DAFREP.txt	Report	Execution report

DPDF - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'     - ACCESS FACILITY PRE-PROCESSING -
'
' ------------------------------------------------------
'
<job id=DPDF>

<script language="VBScript">
Dim MyProc
MyProc = "DPDF"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If



Call Msg_Log (Array("1022" , "DAFD10"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
Call BvpEnv("DAFD10","DAF80",Rep_USR & "\DAF80.txt")
Call BvpEnv("DAFD10","COB80",Rep_USR & "\COB80.txt")
Call BvpEnv("DAFD10","DAFREP",Rep_USR & "\DAFREP.txt")
Call RunCmdLog ("BVDAFD10")

Call Err_Cod(Return , 0 , "DAFD10")


Call Msg_Log (Array("1024"))
'---------------------------
Call DeleteFldr(Rep_TMP)
```

```
Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 19. DUPD - Batch update from DAF tables

DUPD - Introduction

The DUPD procedure performs the batch update of the DSMS Database from a sequential file which mirrors the DAF tables.

Its operating principle is quite similar to that of the DUPT procedure, except for the format of the input transactions.

Execution condition

Refer to the chapter dedicated to DUPT.

Abnormal execution

Refer to the chapter dedicated to DUPT.

DUPD - Input / Processing / Results

USER INPUT

The sequential file of input transactions is produced by a DAF extractor program. Its records mirror the DAF tables (described in the DAF TABLES Manual).

Pos.	Length	Meaning
1	1	Transaction code (C, M, X, D or A, B)
2	10	DAF table code
12	299	DAF table contents (described in the DAF tables Manual).

UPDATE RULES

Update transactions are not sorted.

Each set of transactions impacting a library or session must be preceded by an ASSIGN table code line.

Pos.	Len.	Value	Meaning
2	10	'ASSIGN'	Table code
12	8	uuuuuuuu	User code

Pos.	Len.	Value	Meaning
20	8	pppppppp	Password
28	3	ppp	Product code
31	2	ss	Subsidiary code

PRINTED OUTPUT

Refer to the description of the DUPT output.

RESULT

Refer to the description of the DUPT result.

DUPD - Description of steps

Database integrity check: PDSBAS

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input	Data file
PACDDE	System - Skel. dir. DE	Input	Error message file
PACDRS	User dir.: DSAVRSBAS.txt	Report	Validity report

Return code:

This utility sends a return code 4 and causes an ABEND if the Database is invalid.

Transaction formatting: PDS900

Code	Physical name	Type	Label
PACDGY		Input	Update transactions
PACDIM	Tmp dir.: IM	Output	Formatted transactions

Update of the DSMS Database: PDSUP0

Code	Physical name	Type	Label
PACDDA	DBase dir.: DA	Input Output	Data file

Code	Physical name	Type	Label
PACDDC	DBase dir.: DC	Input Output	VisualAge Pacbase elements file
PACDDX	DBase dir.: DX	Input Output	Cross-references file
PACDDE	System - Skel. dir.: DE	Input	Error messages file
PACDIM	Tmp dir.: WIM.tmp	Input	Update transactions output by DEXP
PACDDJ	Journal dir.: DJ	Input	Journal
PACDRP	User dir.: DUPTRPUP0.txt	Report	Update report

Return codes:

- 0 : No error on files
- 8 : Error on user identification line or parameter
- 12 : Input/output error on a file

DUPD - Execution script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' ------------------------------------------------------
'         - BATCH UPDATE FROM DAF TABLES -
'
' ------------------------------------------------------
'
<job id=DUPD>

<script language="VBScript">
Dim MyProc
MyProc = "DUPD"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Call Msg_Log (Array("1022" , "PDSBAS"))
'-----------------------------------
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
Call BvpEnv("PDSBAS","PACDRS",Rep_USR & "\DUPDRSBAS.txt")
Call RunCmdLog ("BVPDSBAS")
```

```
If Return = 4 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 0 , "PDSBAS")

Call Msg_Log (Array("1022" , "PDS900"))
'-----------------------------------
WshEnv("PACDGY") = Fic_Input
Call BvpEnv("PDS900","PACDIM",Rep_TMP & "\WIM.tmp")
Call RunCmdLog ("BVPDS900")
Call Err_Cod(Return , 0 , "PDS900")

Call Msg_Log (Array("1022" , "PDSUP0"))
'-----------------------------------
Call BvpEnv("PDSUP0","PACDIM",Rep_TMP & "\WIM.tmp")
WshEnv("PACDDE") = Rep_SKEL & "\DE"
WshEnv("PACDDA") =  Rep_BASE & "\DA"
WshEnv("PACDDX") =  Rep_BASE & "\DX"
WshEnv("PACDDC") =  Rep_BASE & "\DC"
WshEnv("PACDDJ") =  Rep_JOURNAL & "\DJ"
Call BvpEnv("PDSUP0","PACDRP",Rep_USR & "\DUPDRPUP0.txt")
Call RunCmdLog ("BVPDSUP0")

Call Err_Cod(Return , 0 , "PDSUP0")


Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 20. DLVB - Replacement of low-values with blanks

The DLVB procedure inserts a blank wherever a low-value is present in the BB Database backup file.

The purpose of this procedure is to make the BB file transferable to various platforms, while avoiding problems due to the presence of low-values during these transfers.

Utilization option

The DLVB procedure gives the user the opportunity to produce a transfer file containing only the 'data'-type

records.

In this case, the backup file obtained on the target platform after transfer will have to be reorganized (DREO procedure) in order to rebuild the cross-references file (DX file).

Execution conditions

None.

DLVB - Parameters / Description of steps

Replacement of low-values with blanks: PDSLVB

Code	Physical name	Type	Label
PACDBB	Save dir.: BB	Input	Database backup
PACDB1	Save dir.: BB.NEW	Output	New Database backup

DLVB - Execution script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE-DSMS
'
' -------------------------------------------------------
'   - CHANGE LOW VALUE CHARACTERS INTO BLANK CHARACTERS -
'
'   OPTION : SUBMIT PROCEDURE WITH PARM='DATA'
'            TO PROCESS DATA OPTION
' -------------------------------------------------------
```

```
'
<job id=DLVB>

<script language="VBScript">
Dim MyProc
MyProc = "DLVB"
</script>

<script language="VBScript" src="Incl.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If



Call Msg_Log (Array("1022" , "PDSLVB"))
'-----------------------------------
Call BvpEnv("PDSLVB","PACDBB",Rep_SAVE & "\OLDBB")
Call BvpEnv("PDSLVB","PACDB1",Rep_SAVE & "\BB-new")

Call RunCmdLog ("BVPDSLVB")
Call Err_Cod(Return , 0 , "PDSLVB")

Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover( Rep_SAVE & "\BB" )


Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)


</script>
</job>
```

Part Number: DEDNT000351A - 7615

Printed in USA