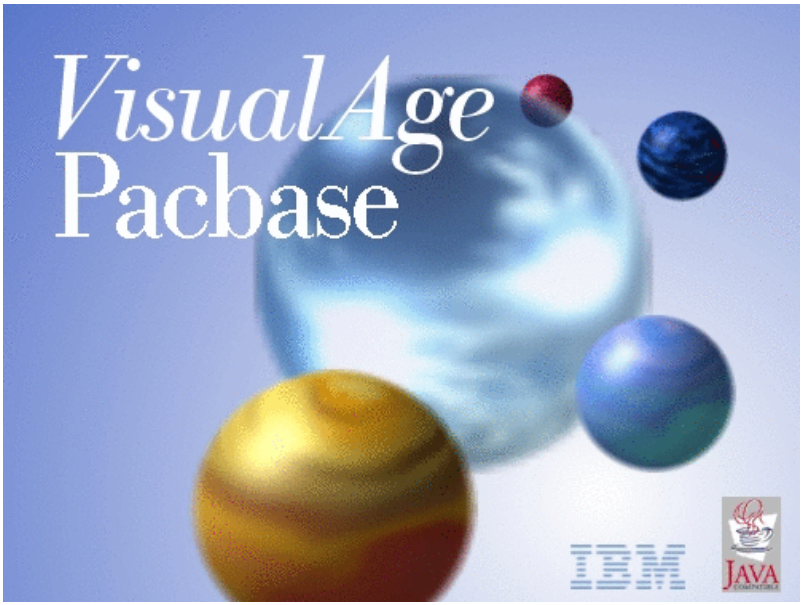
VisualAge Pacbase

DSMS: Installation & Operations UNIX Server

Version 3.5



VisualAge Pacbase

DSMS: Installation & Operations UNIX 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 (May 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 DSMS error messages and HELP documentation,
  - 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

They make up the actual system. They are not impacted by daily transactions, and they must be reloaded each time the system is reinstalled.

These files are:

- the on-line and batch load modules (directory: $DSMSDIR/system/gnt)
- the executable modules (directory: $DSMSDIR/bin)

as well as:

- the DE file which contains the error messages and help documentation of the DSMS function:

Characteristics	Value
Size	About 27,000 records (about 3.5 Mb)
Organization	Indexed
Length	90
Key	17 (position 1)

Characteristics	Value
Location	$DSMSDIR/system/skel directory
Internal name	PACDDE

- The HE file is used to save the screen when the Help function is called in on-line mode.

  This file is located in the $DSMSDIR/data/'db_name'/base directory.

User files

These files contain the data entered by the user and managed by DSMS.

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

- The DSMS data file (DA)

Characteristics	Value
Organization	Indexed
Length	Minimum 80, maximum 350
Key	40 (position 3)
Location	$DSMSDIR/data/'db_name'/base directory
Internal name	PACDDA

- The cross-reference file (DX)

Characteristics	Value
Organization	Indexed
Length	80
Key	50 (position 1)
Location	$DSMSDIR/data/'db_name'/base directory
Internal name	PACDDX

- The VisualAge Pacbase element file (DC)

Characteristics	Value
Organization	Indexed
Length	Minimum 50, maximum 168
Key	31 (position 3)
Location	$DSMSDIR/data/'db_name'/base directory
Internal name	PACDDC

- The DSMS Journal file (DJ)

Characteristics	Value
Organization	Relative
Length	180
Location	$DSMSDIR/data/'db_name'/journal directory
Internal name	PACDDJ

- TP DAF work file (SYSDAF)

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

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

- The Backup file (BB)

Characteristics	Value
Organization	Variable sequential
Length	354
Location	$DSMSDIR/data/'db_name'/save directory
Internal name	PACDBB

- The Journal Archive file (BJ)

Characteristics	Value
Organization	Sequential
Length	180
Location	$DSMSDIR/data/'db_name'/save directory
Internal name	PACDBJ

- The Deactivated Archive file (BQ)

Characteristics	Value
Organization	Sequential
Internal name	PACDBQ

Characteristics	Value
Location	None by default (non-assigned file) . You must assign this file in $DSMSDIR/save/'db_name' if used (see DARC procedure)

The BB and BJ sequential backup files are used as input and output by a number of batch procedures. In this case, they are created and used under two different names: Bx as an input file and Bx.NEW as an output file (BB and BB.NEW for instance). At the end of the procedure (if there is no error) the BxBACKUP.ini file is called. By default, it rotates the processed backup on two copies: Bx is renamed Bx-1, and Bx.NEW is renamed Bx before being deleted.

Note:

- '$DSMSDIR' and 'db_name' are installation parameters.
- '$DSMSDIR' is equivalent to $HOME/dsmsx and 'db_name' is the Database name.

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 query 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 query definition
BVPDSCSI	Batch control on site definition

Chapter 3. Installation

Prerequisite

Hardware and software

- Architecture: A UNIX server
- Memory: RAM memory (64 Mb). You may need additional memory according to the number of servers installed on the same machine.
- Software - Cobol Runtime:
  - 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 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 6.5 million bytes approximately.

System installation

Overall presentation

The product is supplied on a CDROM which contains the following elements:

- dsmsinst.Vnnx,
- DSMSBASE.xxx

('Vnnx' is the number of the version installed).

('xxx' is the current version of the compiler).

CAUTION: Depending on the medium type and the UNIX system, the names of the installation files can be either in uppercase or lowercase letters.

Description of dsmsinst.Vnnx:

The dsmsinst.Vnnx file contains the DSMS (re)installation commands.

These commands are written in shell language.

Description of DSMSBASE.xxx:

The DSMSBASE.xxx files are compressed tar files which contain the DSMS execution and operations files.

Choosing the runtime upon an installation or a reinstallation enables you to install files which are compatible with the runtime in use.

Installation process

It is recommended to backup the system before starting the DSMS installation and to closely follow the various installation steps.

- Update the '/etc/services' file,
- Create a specific login to the product,
- Create or modify the environment variables,
- install the system, from the CD-ROM,
- install the DSMS Database.

Between the system installation and the DSMS Database installation, the Mklink is executed. If its execution ends abnormally, you just have to correct the error and link the executable programs to the runtime if needed. Then, restart the installation of the Database by executing the 'dsmsadmin' procedure (see the 'Installation of the Repository' paragraph in this chapter), which is stored in the following directory:

$DSMSDIR/system/install

WARNING: DSMS must be installed on disks physically present on the UNIX machine and not on NFS-mounted disks.

For the Microfocus or Acucorp versions, the COBOL runtime is required for DSMS to operate properly. It must be installed before DSMS.

WARNING: For the installation of the Acucobol version on HP-UX, you must assign the [nflocks] system variable, 'max Number of File LOCKS', a value higher than 9000.

Updating the '/etc/services' file

Using the 'socket' communication interface for the communication between the server and the clients requires reserving communication port numbers in the '/etc/services' UNIX file.

This file must be updated BEFORE the installation of the DSMS Databases. Since this installation can follow the system installation, this file must be updated BEFORE the system installation.

A communication port number must be associated with each database. Communication ports cannot be chosen at random.

Some ports are reserved for standard 'internet' applications or for later applications. Other ports are allocated in a dynamic way.

So you must choose a port number which is not used in '/etc/services' and which is not being used, i.e. dynamically allocated.

Example:

A port number must be chosen for each on-line server. Let us suppose that we have chosen number 52040.

- The port number 52040/tcp must not be in the '/etc/services' file ;

- The port number must not be currently used. You can make sure of that via the following UNIX command:

```
netstat -an | grep 52040
```

- If both these conditions are met, the UNIX (root) administrator must update the '/etc/services' file using the following line:

```
dsmsx 52040/tcp
```

The port number chosen for each on-line server must be carefully noted because it will be needed when creating DSMS Databases.

Creating a specifix Unix user

Most software running on UNIX must be installed in a specific user's account. This is necessary for security reasons, in order to isolate the various software programs installed on one machine.

NEVER INSTALL UNDER THE "ROOT" UNIX ACCOUNT.

The UNIX (root) administrator must therefore create a UNIX login under which DSMS will be installed.

This login will be the DSMS Adminstrator's.

Modifying the configuration files

Log in with the DSMS administrator login, and update the shell configuration files (.profile, .kshrc or .login).

The environment variables PACDIR and PATH will be created or modified:

```
DSMSDIR="$HOME/dsmsx"
export DSMSDIR
PATH="$DSMSDIR/system/bin:$DSMSDIR/system/proc
     :$COBDIR/bin:$PATH"
export PATH
```

The environment variables COBDIR, COBPATH, ACUDIR and, CODE_PREFIX must be added, according to the Run-Time:

For Microfocus :

```
COBDIR=<COBOL Run-Time directory path>
export COBDIR
COBPATH="$DSMSDIR/system/gnt"
export COBPATH
```

For AcuCobol:

```
ACUDIR=<COBOL Run-Time directory path>
export ACUDIR
CODE_PREFIX="$DSMSDIR/system/acu"
export CODE_PREFIX
```

If the TMPDIR variable (temporary directory of the Cobol Run-Time) is assigned, the specified directory must exist.

For the COMPAQ/TRUE64 (DIGITAL), SUN, HP-UX or LINUX versions of Microfocus, you must create or modify the LD_LIBRARY_PATH variable:

```
LD_LIBRARY_PATH=/usr/lib:$DSMSDIR/coblib:$DSMSDIR/system/bin
   export LD_LIBRARY_PATH
```

If you use Server Express, LD_LIBRARY_PATH must contain:

```
LD_LIBRARY_PATH=/usr/lib:$COBDIR/lib:$DSMSDIR/system/bin
   export LD_LIBRARY_PATH
```

If you use Acucobol, LD_LIBRARY_PATH must contain:

```
LD_LIBRARY_PATH=/usr/lib:$ACUDIR/lib:$DSMSDIR/system/bin
   export LD_LIBRARY_PATH
```

For the AIX version, you must use the LIBPATH variable instead of LD_LIBRARY_PATH.

For the HP-UX version, in addition to the LD_LIBRARY_PATH variable, you must set the SHLIB_PATH variable as following:

```
export SHLIB_PATH=$DSMSDIR/system/bin:$SHLIB_PATH
```

The COBOL Run-time corresponds to the directory which contains the executable files and libraries required to execute COBOL programs.

For Microfocus, COBDIR can generally take the following values: /usr/lib/cobol, /usr/lpp/cobol or /opt/cobol. (For further details on the COBDIR environment variable, refer to the installation documentation of your COBOL compiler and to your COBOL manuals.)

For Acucobol, the ACUDIR variable contains the compiler or the Cobol Run-Time installation directory.

Once you have performed these updates, restart the configuration initialization by typing '. .profile', and use the UNIX command 'set' to check that the modifications have been taken into account in the configuration files.

If the environment variables are not correctly initialized, log out and then log in again, in the DSMS administrator's account.

Defining the code page

The Repository code page is an IBM-923 code page equivalent to the ISO8859-1 code page which supports the Euro character (ISO8859-15).

Positioning the 'LANG' code for the UNIX user

In order to visualize properly the reports which contain national characters, the user must define a code page compatible with the DSMS Repository code page.

Refer to your UNIX documentation to define a code page compatible with the ISO8859-1 or ISO8859-15 code page.

As a general rule, positioning the 'LANG' variable alone is enough to define the code page to be used.

Example : for the support of French national characters, you set:

- on AIX, SOLARIS, OSF1 :

```
export LANG=fr_FR.ISO8859-1
```

- on HP-UX :

```
export LANG=fr_FR.iso8859-1
```

Installing from a CDROM

If necessary, mount the CDROM disk driver on a system directory.

launching the installation procedure

To ensure a correct installation of the software, check that the disk space is equal to five times the size of the DSMSBASE.xxx file in the installation directory.

After downloading to the DSMS administrator's login directory, or the CDROM mounting, go to the following directory:

```
ID350'package_code'.'version_code'
```

which is located in $HOME or in the CDROM mounting directory.

Example on AIX:

```
cd $HOME/ID350AIX.Vnn (nn = version number)
```

and type the following command:

```
sh dsmsinst.Vnnx (nn = version number,
                  x = sub-version number)
```

followed by a carriage-return (Enter or Return key).

The installation is made up of the following steps:

- Check the consistency of the environment variables
- Display the installation menu
- Select the COBOL run-time in use (if relevant)
- Select the generation language code
- Create the installation directory
- Distribute the files coming from DSMSBASE.xxx
- Create the standard test Database
- Create the journal_dsmsinstall file.

Check the consistency of the environment variables

Before running dsmsinst.Vnnx, the DSMS Administrator must check the modifications done in the configuration file, even though the installation process performs some consistency checks.

Display the installation menu

The installation menu looks like this:

```
*********************************************************
                          DSMS
*********************************************************
 inst            :  installation of this version
*********************************************************
Type the command name, or 'x' to quit dsmsinst.Vnnx:
```

Description of commands:

- The 'inst' command starts the installation process.

Type the command and follow the instructions given by dsmsinst.Vnnx.

Choose the Cobol run-time in use

This choice is proposed on some platforms upon installation.

Ask your UNIX Administrator to know which run-time is used.

Select the generation language

The generation language code is 'en' for English or 'fr' for French.

Create the installation directory

dsmsinst.Vnnx requires the installation directory to be named 'dsmsx' in order to:

- isolate DSMS from the other applications and so control its evolution more easily,
- allow dsmsinst.Vnnx to control the type of processing (installation or re-installation) to be done.

Distribute the files

During this step, dsmsinst.Vnnx creates the directories described further on in this manual (paragraph 'Description of created Directories') and extracts all the DSMSBASE.xxx files.

Create and name the Test Database

The test Database must be given a name at installation time. You enter this name in an interactive manner, by answering questions about the Database (4 alphanumeric characters maximum).

This name allows you to create the Database's specific sub-directories:

```
- $DSMSDIR/assign/'db_name'
- $DSMSDIR/bases/'db_name'
- $DSMSDIR/input/'db_name'
- $DSMSDIR/journal/'db_name'
- $DSMSDIR/save/'db_name'
- $DSMSDIR/tmp/'db_name'
```

Select the DC file access path

For the sites which operate a VA Pac Database, it allows to assign the DC file in this Database (usually $PACDIR/data/'db_name'/bases)

Execute the batch initialization procedures

The batch procedure that initializes the test Database is DRST (Loading of the test database).

NOTE: The Micro Focus COBOL run-time is required to run the batch procedures, as well as to start the batch and on-line servers.

Update the journal_dsmsinstall file:

The journal_dsmsinstall file contains some information resulting from the use of dsmsinst. After an installation, it contains the following information:

- Installation date
- Release number
- Version number
- Name of the character set used
- Test Database creation date.

Complement: Installing DAF environment

The principle of the DSMS ACCESS FACILITY (DAF) function implies transforming the DSMS Database access SQL queries, written in user programs, via the generation of data and calls to sub-programs 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.gnt, installed in the $DSMSDIR/system/gnt directory.

The user can use the DPDF procedure to process his/her generated programs which use 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 the 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 (.gnt files) when delivered, in the $DSMSDIR/system/gnt 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 use the DAF function. 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

Inserting the 'DAF dictionary' into VisualAge Pacbase via the UPDT batch update procedure is 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 PROGRAMS

The $DSMSDIR/daf directory contains a sample compilation script and a sample execution script for a DAF program ('dafcomp' and 'dafrun').

It is advised to duplicate the directory in a work directory, for example 'dafuser', via the command:

cp -r $DSMSDIR/daf /dafuser

And then copy the DAF generated programs to the work directory.

Go to the work directory to modify and launch the compilation and execution scripts.

The compilation script may be used to compile the DAF system sub-programs.

WARNING: the sources of the DAF generated programs must not be modified under the editor after the operation of the pre-processor because the editor removes the 'low values' generated by the pre-processor.

EXECUTION OF A DAF EXTRACTOR

The assignment of the following files must precede the execution of the DAF extractor:

- Permanent input files:
  - DSMS Data file : PACDDA
  - VA Pac elements file : PACDDC
  - Cross-references files : PACDDX
  - Error message file : PACDDE
- DAF work file : SYSDAF
- User files if relevant.

Sample compilation script

```
#!/bin/sh
#
# COMPILATION EXAMPLE OF DAF PROGRAMS
# -----------------------------------
#
# (compilation of PGDAF et PGDAFP programs)

# Cobol compiler assignment :

COBDIR=/usr/lib/cobol
export COBDIR

# PATH assignment :

PATH=$COBDIR/bin:$PATH
export PATH

# Compiler directives :

COBOPT="-C ASSIGN=EXTERNAL -C NATIVE=ASCII -C SEQUENTIAL=LINE"
COBOPT="$COBOPT -C PERFORM-TYPE=OSVS -C OSVS"
COBOPT="$COBOPT -C NOBOUND -C IBMCOMP"
COBOPT="$COBOPT -C NESTCALL -C DEFAULTBYTE=32"

# Program list :
```

```
PGM="PGDAF.cbl PGDAFP.cbl"

# Start compilation :

cob -uv $PGM $COBOPT

# Compiled files = PGDAF.gnt et PGDAFP.gnt
```

Sample execution script

```
#!/bin/sh
#
# EXEMPLE OF AN EXECUTION PROCEDURE USING A DAF PROGRAM
# -----------------------------------------------------
#
# (execution of PGDAF program)

# Cobol compiler assignment :

COBDIR=/usr/lib/cobol
export COBDIR

# Path assignment :

PATH=.:$COBDIR/bin:$PATH
export PATH

# DSMS directory assignment :

DSMSDIR="/dsms200/dsmsx"
export DSMSDIR

# COBPATH assignment :
# current directory + DSMS BATCH programs directory

COBPATH=.:$DSMSDIR/batch/gnt
export COBPATH

# DSMS database name (mandatory) :

BASE=test

# DAF assignment (mandatory) :

. $DSMSDIR/assign/$BASE/PACDDE.ini
. $DSMSDIR/assign/$BASE/PACDDC.ini
. $DSMSDIR/assign/$BASE/PACDDA.ini
. $DSMSDIR/assign/$BASE/PACDDX.ini
SYSDAF=./wdaf
export SYSDAF

# Users files assignment :

FILE1=./file1
```

```
export FILE1

# Start execution :

cobrun PGDAF

# Deletion of the temporary files :

if [ -r "$SYSDAF" ]
then
 rm $SYSDAF*
fi
```

List of installed elements

The installation copies:

- The conversion file, the file of the error messages output by the procedures when an error is detected or only for information
- The programs
- The procedures
- The Database creation utility
- The start-up sample scripts.

After the installation, the following directories are created:

- $DSMSDIR/config,
- $DSMSDIR/data,
- $DSMSDIR/system.

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 $PACDIR/data/'db_name'/base 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 PACDIR/data/'db_name'/base directory of the VA Pac Database to be monitored.

To relocate DC, change the assignment file of DC in both the DSMS and VA Pac installations. This assignment file is in the $PACDIR/config/base directory of the VA Pac installations and in the $DSMSDIR/config/base directory of the DSMS installation.

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 $PACDIR/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 $PACDIR/data/'db_name'/base directory of the VA Pac Database to be monitored.

To relocate DC, change the assignmentfile of DC in both the DSMS and VA Pac installations. This file is in the config\'db_name' directory of the DSMS and VA Pac installations.

Usability 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 stopped.

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

A system element: the online server

When started up, each listener executes a command file (such as 'BVPSERVER.ini') to assign the environment variable it needs.

This file is created upon the creation of the listener in the $DSMSDIR/config/'db_name' directory. It contains, among others, the following environment variables:

```
- BVPSOCKET        : port number (socket),
- SRV_DIR          : directory which contains the trace
                     files of the listener.
```

The listener, whose executables (dstp, dsserver, dslaunch) are in the $DSMSDIR/system/bin directory allows to:

- Set the listener(s) in an active or inactive mode,
- Supply information concerning the listener(s),
- Purge workstation(s) attached to a listener,
- Purge the listener(s).

To carry out the operations listed above, you must run the interpreter of the listener commands (dstp).

The interpreter can be run in two different modes:

- the 'command' mode, for which you type:

  ```
  dstp <command>
  ```

- the 'shell' mode, for which you type :

  ```
  dstp -s
  ```

The main benefit of the command mode is that it allows the insertion of the listener's commands in a command file.

For example, the com_dsms file contains the following commands:

```
# display of the listeners' status
dstp info
# start-up of the DTST listener
dstp start DTST
# 10-second display of information on the DTST listener
dstp info DTST
# submission of the purge command for workstation 003
connected to server DTST
dstp purge DTST 003
```

The main benefit of the 'shell' mode is that it avoids running the interpreter for each command (the interpreter remains pending for the next command).

The following commands are available:

```
-debug           activation/de-activation of the debug mode
```

```
-exit            exit shell mode
-help            help on the specified command
-info            information on the specified listener(s)
-purge           purge of workstation
-purge_server    purge of listener
-shutdown        listener shutdown without confirmation
-start           start-up of specified listener
-stop            listener shutdown with confirmation
-view            display of listener status
```

A detailed description of the commands is provided in Subchapter 'Description of Commands'.

Execution conditions

The DSMSDIR environment variable must be initialized.

The COBPATH environment variable must contain the access path to the on-line modules '$DSMSDIR/system/gnt'.

See Chapter 'Installation', Subchapter 'System Installation' - 'Modifying the Configuration files'.

Debug command

This command activates or de-activates the debug mode on the listener. The listener's name must be entered as a parameter, followed by 'on' for activation or 'off' for de-activation.

The result files are:

```
srv[process_number].txt
```

to trace the listener which is listening to new connections,

```
dial[process_number].txt
```

to trace each connection to the listener.

So there is a trace for each connection to the listener.

These files are located in the directory specified by the SRV_DIR environment variable assigned in the configuration file '$DSMSDIR/config/'db_name'/BVPSERVER.ini'. Its default value is: '$DSMSDIR/data/"nom_base"/tmp/server'.

Example: Starting the debug mode on a listener named DTST:

```
dstp debug DTST on       (command mode)
DSMS : debug DTST on     (shell mode)
```

Example: Stopping the debug mode on the DTST listener:

```
dstp debug DTST off      (command mode)
DSMS : debug DTST off    (shell 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.

  The 'debug on' command implements a trace level 1 on an active listener. To activate another trace level, you must set the SRV_TRACE variable in 'BVPSERVER.ini' 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

The 'debug off' command stops the production of a trace for the new connections to the listener.

Exit command

With this command, you exit the 'shell' mode (command interpreter) previously activated via the dstp -s command.

Help command

This command enables you to display help on an administration command of the listener. If no parameter is specified, the list of all the available commands is displayed. If a specific command is indicated, help on this command is displayed.

Example: Display requested for the 'start' command syntax

```
dstp help start          (command mode)
DSMS : help start        (shell mode)
```

Info command

This command enables you to display information on the listener(s). Followed by the Database name (info 'db_name') it displays the following information:

```
- active  (if the listener is active)
- not active (if the listener is not active)
- Error (if the listener has stopped abnormally).
```

Purge command

This command enables you to purge a workstation (in other words, to interrupt a workstation's connection).

If the dstp interpreter is in command mode, the syntax of the purge command is as follows:

```
'dstp purge 'db_name' <workstation number>'
```

In 'shell' mode, the purge command can have two syntaxes:

- 'purge <workstation number>'
  if the prompt is not 'DSMS :'.
  (The prompt takes the listener's name as the value when, for instance, the 'view' command has been used.)
- 'purge 'db_name' <workstation number>'
  if the prompt is 'DSMS :'.

Purge_server command

This command enables you to purge a listener in the case of an abnormal execution, viewed as an 'error' via the 'info' command.

In command mode, this command has the following syntax:

```
dstp purge_server 'db_name'.
```

In shell mode, it has the following syntax:

```
purge_server 'db_name'.
```

NOTE:

This command removes the listener process(es) as well as the IPC resources in use (semaphores, shared memory).

Shutdown command

This command enables you to shutdown the listener. The Database name must be entered as a parameter of the command.

Example: shutting down the listener named DTST:

```
dstp shutdown DTST       (command mode)
DSMS : shutdown DTST     (shell mode)
```

Start command

This command enables you to activate a listener. The Database name must be entered as a parameter. The BVPSERVER.ini initialization file must exist in the $DSMSDIR/config/'db_name' directory. The listener start-up program, dslaunch, starts and activates the dsserver process. If a problem occurs (listener jam), you must first try the following command:

```
dstp purge_server 'listener_name'.
```

If this command is not effective, you can kill the dsserver process using the 'Kill -15' or 'Kill -9' command followed by the process number (PID).

Example: activation of the listener named DTST:

```
dstp start DTST          (command mode)
DSMS : start DTST        (shell mode)
```

Stop command

This command stops a listener. The Database name must be entered as a parameter. You will be asked to confirm the interruption of the server.

Example: shutdown of the listener named DTST.

```
dstp stop DTST           (command mode)
DSMS : stop DTST         (shell mode)
```

View command

This command enables you to display information about a listener on a given Database. The list of the connected workstations, the IP address of the client workstation, the PID number of the client process launched by the listener, the name of the COBOL program executed and its 'elapsed' execution time in milliseconds as well as information about the semaphore status allow the synchronization of the concurrent accesses to the Database.

Example: display of information about the DTST listener:

```
dstp view DTST           (command mode)
DSMS : view DTST         (shell mode)
```

Installing the repository

DSMS Database

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

A DSMS Database can be installed right after the system installation, or later, via the 'dsmsadmin' procedure which is located in the directory:

```
$DSMSDIR/system/install.
```

This installation involves the following steps:

Displaying the installation menu

The installation menu looks like this:

```
************************************************************
                      DSMS
************************************************************
 crebase     :  creation of a new database
************************************************************
Type the command name or 'x' to exit dsmsadmin :
```

Description of commands:

- The 'crebase' command starts up the installation process.

Enter the command and follow the instructions given by dsmsadmin.

Entering the Database code (except for the test Database)

Enter the Database code on 4 characters and its name on 30 characters (alphanumeric uppercase characters).

When installing the first Database (test Database), the Database code is always DTST.

Selecting the Database language code

This code can be:

- en (English)
- fr (French)

Entering the socket number allocated to the listener

The socket number is a 5-digit number between 49152 and 65535.

It must be declared in the /etc/services file. (see paragraph 'Updating the '/etc/services' File').

Creating the Database subdirectories

- $DSMSDIR/data/DTST/base
- $DSMSDIR/data/DTST/save
- $DSMSDIR/data/DTST/script
- $DSMSDIR/data/DTST/tmp
- $DSMSDIR/data/DTST/users
- $DSMSDIR/config/DTST

Installing the test Database

The test Database is supplied as a BB backup in the $DSMSDIR/system/install/base/save directory. It is used upon the Database restoration via the DRST procedure.

Updating the journal_dsmsinstall file

Addition, in the journal_dsmsinstall file, of the information related to the Database creation:

- Database name and creation date.

Deleting a Database

To delete a Database, you must go to the installation user's login directory to delete the following directories:

- $DSMSDIR/data/[db_name]
- $DSMSDIR/config/[db_name]

You must enter the following command: rm -r [Directory_name]

Connection

The listener must be started up for terminals to be able to connect to DSMS.

You can access an on-line server via a 3270 emulator.

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

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

- 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 when the Database is created.

When the emulator is started up, the system automatically positions to the DSMS transaction.

Chapter 4. Server reinstallation

Reinstallation

Overall presentation

The system part of DSMS must be re-installed when a new sub-version comes out, with corrections and/or new specific developments to the current version.

Generally, only the error message system file and the programs are impacted by the new sub-version.

General notes

- The re-installation procedure does not create the directories, which are supposed to be identical to those created during the first installation of the version.
- It does not copy the batch procedures ('$DSMSDIR/system/proc' directory) if they have been adapted to the site by the Database administrator, except if the new sub-version cannot run with the old procedures.

  The new procedures are copied to the directory:

  ```
  $DSMSDIR/system/proc.Vnn (nn = version number)
  ```
- The operational startup scripts, located in the $DSMSDIR/data/[db_name]/script directory, are not impacted. To get the latest version of the scripts, upon the re-installation, refer to the startup script models in the /system/install/basefra or baseeng/script directory.

Launching the reinstallation procedure

WARNING: The DSMS Database server must be stopped.

After downloading to the DSMS administrator's login directory, or the CDROM mounting, go to the following directory:

```
ID350"hardware_code"."version_code"
```

as in an installation operation (see sub-chapter 'DSMS Installation'), and type the command:

```
sh dsmsinst.Vnnx (nn = version number,
                  x = sub-version number)
```

followed by a carriage return (Enter or Return key).

Description of steps

The re-installation procedure includes the following steps:

- Check the consistency of the environment variables (.profile, .kshrc or .login file),
- Display the re-installation menu,
- Choose the COBOL run-time in use (if necessary),
- Distribute the files coming from DSMSBASE.xxx,
- Update the journal_dsmsinstall file.

Check the consistency of the environment variables

See subchapter 'Installation'.

Display the re-installation menu

The re-installation menu looks like this:

```
************************************************************
               DSMS
************************************************************
 reinst       :  version re-installation
************************************************************
Type the command name or 'x' to exit dsmsinst.Vnnx:
```

Description of commands

- You perform an overall re-installation via the 'reinst' command.

Enter the command and follow the instructions given by dsmsinst.Vnnx.

Choice of Cobol run-time in use

See subchapter 'Installation'.

Update the journal_dsmsinstall file

Addition, to the journal_dsmsinstall file, of information about the re-installation:

- Re-installation date,
- Release number,
- Version number.

Chapter 5. Utility sub-programs

Presentation of the utilities

Conversion utilities

```
cgiux2dos  : for the conversion of a UNIX-format file
             into a DOS-format file
cgidos2ux  : for the conversion of a DOS-format file
             into a UNIX-format file
cgitrans   : for the conversion of HP, iso8859, and
             pc850 characters
```

System utilities

```
dsdate     : gives the dates of system files and
             programs
```

All the utility programs are found in the $DSMSDIR/bin directory.

To know how they operate, type the following command:

```
<utility name> -h
```

Example :

```
cgitrans -h
```

Chapter 6. Retrieval of a 2.n version

Overall presentation

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

- save the 2.n version,
- reorganize the 2.n version,
- restore in the new version's environment, using the file produced by the previous reorganization.

Chapter 7. 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 the cross-reference 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 (DUPT, DUPD), of Events, Changes, Sites or Tables.
- Pre-processing of DAF source files (DPDF).
- Renaming of Table, Site and Keyword codes (DREN).

RETRIEVAL OF A DATABASE ONTO 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 an error code, described in a message displayed on the screen.

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
```

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

Status	Message
23	Record not found
24	Boundary violation
30	System error
34	Boundary violation (sequential)
35	File not found
46	No current record (for a READ). The error occurs when the previous operation is an abended START, which left the pointer not defined.
48	Attempt at writing on a file which is not open or on a sequential file open in I/O.
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 various procedures are executed in sequence.

(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 submission

The command files of the procedures are created under the $DSMSDIR/system/proc directory during the installation step.

To run a procedure, you can:

- Directly execute the command file of the batch procedure followed by these parameters:

  ```
  procedure "database code"-i "User input file" +t
  -u "user directory" -t "temporary directory"
  ```

- or execute the batch procedure via a start-up script:

  This script, written in UNIX shell, sets the environment variables (optionally, the user input) and executes the command file of the procedure.

  A sample operational script is supplied for most of the procedures and for each Database created, in the following directory:

  ```
  $DSMSIR/data/"database code"/script
  ```

In any case, the user input supplied must be checked so that it conforms to your environment.

Structure of a procedure

The Database Manager must sometimes modify the command files of the batch procedures.

For example, if he/she wishes to save the files on two different disks or simply move them, the resulting changes in the command files might be very important.

This is the reason why procedures are designed in such a way that the standard installation can be easily modified and that changes to fit the operating constraints are limited.

The purpose of this subchapter is to analyze a batch procedure in order to explain how it works and to help you in the fitting process.

Parameters

- The Database code (4 characters):

  It is required.

- The complete name of the user input file:

  It is required when the procedure is directly executed.

- Parameter ″+t″:

  It is optional and is used to prevent the default clearing of temporary files.

- User directory:

  It is optional and it is used to change the user directory default assignment.
- Temporary file:

  It is optional and it is used to change the default assignment of the temporary files directory.

Environment variables

- BVPINPUT:

  This variable contains the user input and is assigned as follows:

```
BVPINPUT=`cat <<eof
1rst user line
2nd  user line
   .
   .
eof`
export BVPINPUT
```

  If the : $ ` ' " characters are used, they must be preceded by two \.

  This variable is not effective if the -i parameter is used.
- DSUTI:

  This variable contains the user code, which will be used to assign the "users" and "tmp" directories. It is set by default with the DSMS user code in the user input.

  It is required if the user input is not filled in or if it does not include any DSMS user code.

  The assignment process is made as follows:

```
DSUTI="user code"
export DSUTI
```
- DSBASE:

  This variable contains the Database code, which replaces the code entered as a parameter of the procedure.

  The assignment process is made as follows:

```
DSBASE="database code"
export DSBASE
```
- NOBVPPAUSE:

  If it is set to "yes", this variable inhibits any pause during the running of the procedure when information messages are displayed.

  The assignment process is made as follows:

```
NOBVPPAUSE="yes"
export NOBVPPAUSE
```
- NOBVPERR:

If it is set to ″yes″, this variable inhibits any pause in the running of the procedure when error messages are displayed.

The assignment process is made as follows:

```
NOBVPERR="yes"
export NOBVPERR
```

- ″procedure″_INPUT:

  This variable enables you to indicate the full path (directory and name) of the file containing the user input.

  The assignment process is made as follows:

```
"procedure"_INPUT="directory/file"
export "procedure"_INPUT
```

- Some environment variables are also used to change the default assignment of the temporary files and of user files produced (reports or output files), either throughout the whole procedure, or only during one step in the procedure execution.

  The assignment process is made as follows:

```
"procedure"_"file code"="directory/file"
export "procedure"_"file code"
```

or

```
"step"_"file code"="directory/file"
export "step"_"file code"
```

Step names and file codes are described in the 'Description of Steps' section for each procedure.

Display and check of parameters

The execution of a procedure starts with the execution of the command file:

```
. $DSMSDIR/system/proc/DSINIT.ini
```

This file is created upon installation in the $DSMSDIR/system/proc directory. It controls the parameters of the procedure.

If it detects an error, DSINIT.ini displays the corresponding error message and stops the procedure with a return code equal to 20.

If it does not detect any error, the procedure then displays the directories assignments.

In order for you to view these assignments, at least during installation tests, the execution stops momentarily with the following message:

```
**********   Check your parameters   *********
Press Control_C to stop the execution
Press Return to carry on
```

If you do not want to stop the execution momentarily, you must set the NOBVPPAUSE environment variable to 'yes'.

Assignment and coding of files

Each step must be assigned the adequate files.

- THE DATABASE FILES

You assign these files by calling the command files, created upon installation in the directory:

```
$DSMSIR/config/"database_name".
```

Example of the assignment of the DA file:

```
. $DSMSDIR/config/$1/PAC7DA.ini
```

The main interest in these files is to centralize the assignment of each Database file in a single place.

The user who wants to modify the standard location of a file only has to adapt the assignment file.

Note: the same files are used when the listeners are started up.

- THE BACKUP FILES

These files are assigned by calling the commands files, created upon installation in the directory:

```
$DSMSDIR/config/"database_name".
```

Example of the assignment of the BB file:

```
. $DSMSDIR/config/$1/PACSAVBB.ini
```

By default, the BB file is located in $DSMSDIR/data/$1/save.

The names of the backup files used by batch procedures are standardized:

```
input back-up file (read) = BB
output back-up file (created by the procedure) = BB.NEW
```

This simplifies the management of these files (see for example the 'Back-up files Management' section a little further on).

- OUTPUT REPORTS AND FILES

The location of output reports and files is determined by a call to the PACUSERS.ini command file:

```
. $DSMSDIR/config/$1/PACUSERS.ini
```

This file is created when a Database is created in the directory:

```
$DSMSDIR/config/'database_name'.
```

It contains:

```
# Command file for assignment of DSMSUSERS environment
variable
# ( 'users' directory )
# Description of parameters : $1      = database name
#                             $BVPUTI = DSMS user code
DSUSERS=$DSMsDIR/data/$1/users/$DSUTI
export DSUSERS
```

Using the -u parameter replaces this default assignment.

When a procedure is executed, a subdirectory named ″procedure code″_″process number″ is created in the $DSUSERS directory.

For the DPRT procedure, the process number is replaced by the job number.

The names of the output reports start with the code of the procedure which outputs them.

More precisely, the reports are coded on nine characters plus an extension (.txt), in the following manner:

- the first four characters correspond to the procedure code,
- the next two correspond to the last two characters of the file (RU in PACDRU),
- the last three characters correspond to the last three characters of the program code (380 in PDS380).

Example: DARC procedure, PDS380 program

```
PACDRU report     --> DARCRU380.txt
```

For the coding of the result files, refer to the 'Description of steps' section of each procedure.

4. Parameters assignment and coding

For each step, the right files must be assigned.

- DATABASE FILES

These assignments are performed via the call to command files, created upon installation in the following directory:

```
$DSMSDIR/assign/'db_name'.
```

For instance, the assignment of the DE file is:

```
$DSMSDIR/assign/$1/PACDDE.ini
```

The command files' main purpose is to put in one single place each Database file assignment. To modify the standard location of a file, only the assignment file needs to be adapted.

Note: the same files are used when starting up the servers.

- BACKUP FILES

As for Database files, the assignments are made through the call to command files created upon installation in the directory:

```
$DSMSDIR/assign/'db_name'.
```

Example: assignment of the BB file:

```
$DSMSDIR/assign/$1/PACSAVBB.ini
```

By default, the BB and BJ backup files are located in the directory:

```
$DSMSDIR/save/'db_name'.
```

For all the batch procedures which use backup files, file names are standardized:

```
Input backup (read) = Bx
Output backup (created by the procedure) = Bx.NEW
```

This way, file management is made easier (for instance, see Subchapter 'Backup file management').

- TRANSACTION FILES

All the transaction files that are used as procedure input have the following name format: MBxxxx (where xxxx is the procedure's name).

All the procedures' output transaction files have the following name format: MVxxxx (where xxxx is the procedure's name). These files can contain, for instance, transactions generated by extraction procedures.

The transaction file location is determined by the PACINPUT environment variable, positioned in every procedure by a call to the PACINPUT.ini command file:

```
$DSMSDIR/assign/$1/PACINPUT.ini
```

The PACINPUT.ini file is created upon installation or upon a Database creation, in the directory:

```
$DSMSDIR/assign/'db_name'
```

It contains:

```
# PACINPUT environment variable assignment script
# (directory: input)
# Description of parameters: $0      = procedure name
#                            $1      = database name
#                            $PACRAD = file radical
#                            $PACSUF = directory extension
PACINPUT=$DSMSDIR/input$PACSUF/$1/$PACRAD
export PACINPUT
```

Example of assignment in the DEXT procedure:

```
PACDMB=$PACINPUT'MBDEXT'
export $PACDMB
PACDIM=$PACINPUT'MVDEXT'
export $PACDIM
```

- OUTPUT REPORTS

All the procedures' output reports are stored in the temporary files directory and their names start with the code of the procedure that produces them.

For details on report name coding, refer to Subchapter 'Output Reports'.

Recommendation on use

The objective of this subchapter is to make the person in charge of the Database aware of the specificities of the DSMS procedures executed on the UNIX system.

General remarks

1. Each procedure must be passed parameters. All the parameters which may be called in a procedure must be present, even if they are not actually used.
2. When user input is expected in a procedure, even if it is optional, the corresponding transaction file must be present when the procedure is being executed.

   For a user input directly entered in the script, if the : $ ` ' " characters are used, they must be preceded by two \.
3. No protection is guaranteed if a BATCH procedure updating the Database system or evolving files is started up while users are interactively

updating these same files. One person (the Database manager) must be able to start up the batch procedures which update the Database. He/she must therefore ensure the protection of the Database data (by closing the on-line servers for example).

4. The temporary work files created by the batch procedures are automatically destroyed at the end of the procedure, except if there was an abend and if a return code other than 0 is sent.
5. The batch procedures must be executed from the UNIX machine.
6. The presence of special characters in the entities code is NOT recommended. The EURO character, for example, is source of problems on ACU.

Management of temporary files

For each procedure you should consult the corresponding chapter for a detailed description of these files.

In all cases, enough disk space should be freed in the chosen user directory to ensure that the procedure runs smoothly.

Temporary sort files:

When a program executes a sort, the called COBOL routines also use a temporary file independent of those mentioned above.

This file is created by default in the /usr/tmp directory.

Its size can be 3 or 4 times the size of the file to be sorted.

If the default directory is too small, the TMPDIR directory assigns another directory for the temporary sort files:

```
TMPDIR=/tmp2
export TMPDIR
```

Management of backup files

All the procedures which create one of the backups call a command file if they end without error.

These files are in the $DSMSDIR/config/"database_name" directory and are named BxBACKUP.ini (where x = B for the BB backup of the DSMS Database, x = J for the BJ backup of the DSMS journal). They are created when the Database is created and contain:

```
# Script for the rotation of the Database backup files
. $DSMSDIR/config/DTST/PACSAVBx.ini
if [ -f "$PACSAVBx" ]
```

```
then
   mv -f $PACSAVBx $PACSAVBx'-1'
fi
mv -f $PACSAVBxNEW $PACSAVBx
```

Characteristics of the BxBACKUP file:

- proceeds by 'mv' to avoid copies of the backup files (these copies may take a long time),
- guarantees that the Bx file is definitely the last backup.

These files do not claim to cover all the operation constraints of all sites. The Database manager generally has to adapt them, taking the characteristics above into account.

Chapter 8. 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 the case of an error on one of the options, an error message is sent 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 the DARC, and even the DRST procedures cannot be executed then.

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.

FIRST ARCHIVING OF THE DATABASE

So that the first archiving process ends normally, the BJ file of archived transactions, used as input to the procedure, is provided as an empty file in the database SAVE directory.

DEACTIVATION OF THE ARCHIVED TRANSACTIONS

Two situations are possible for the deactivation of the transaction file:

- The BJ file's deactivated archive is not to be kept: The PACDBQ (internal name) file must be assigned as '/dev/null' (as a default, this is done in the procedure's command file).
- The BJ file's deactivated archive is to be kept: the PACDBQ (internal name) file must be assigned and must correspond to a file on the disk. The procedure's command file must be modified (example: "PACDBQ=$DSMSDIR/save/BQ").

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
PACDDJ	Journal dir.: DJ	Output	Journal file to be reinitialized
PACDRU	User dir.: DARCRU320.txt	Report	Reinitialization report

DARC - Execution script

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DARC BATCH PROCEDURE
# * ------------------------------------------------------
# *      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.
# *
# * ------------------------------------------------------
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DARC"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
STATUS=`dstp info $1 | grep "Server Status" | cut -d: -f2`
if [ "$STATUS" != " Inactive" -a "$STATUS" != "" ]
then
   DSMSG 1012 "DARC"
   DSMSG 1037 $1
   DSERR
   exit 12
fi
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDJ.ini
. $DSMSDIR/config/$1/PACSAVBJ.ini
PACDJB=$PACSAVBJ
export PACDJB
PACDBJ=$PACSAVBJNEW
export PACDBJ
PACDMB=$DSINPUT
export PACDMB
PACDBQ=/dev/null
export PACDBQ
PACDRU=`DSENV PDS300 PACDRU $DSUSERS/DARCRU300.txt`
export PACDRU
DSMSG 1009 "BVPDS300"
rtsds BVPDS300
RETURN=$?
case $RETURN in
 0)
   . $DSMSDIR/config/$1/PACDDE.ini
   . $DSMSDIR/config/$1/PACDDA.ini
   . $DSMSDIR/config/$1/PACDDJ.ini
   PACDMB=$DSINPUT
   export PACDMB
   PACDRU=`DSENV PDS320 PACDRU $DSUSERS/DARCRU320.txt`
   export PACDRU
   DSMSG 1009 "BVPDS320"
   rtsds BVPDS320
```

```
   RETURN=$?
   case $RETURN in
    0)
      ;;
    *)
      DSMSG 1012 "BVPDS320"
      DSMSG 1025
      DSERR
      DSRMTMP
      exit $RETURN
      ;;
    esac
   ;;
12)
   DSMSG 1012 "BVPDS300"
   DSMSG 1018
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
 *)
   DSMSG 1012 "BVPDS300"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSMSG 1016 "BJBACKUP.ini"
sh $DSMSDIR/config/$1/BJBACKUP.ini
DSRMTMP
exit $RETURN
```

Chapter 9. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DPRT BATCH PROCEDURE
# * ------------------------------------------------------
# *      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
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DPRT"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ***************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDC.ini
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDX.ini
PACDMB=$DSINPUT
export PACDMB
PACDIA=`DSENV PDSB PACDIA $DSUSERS/DPRTDA.txt`
export PACDIA
PACDIB=`DSENV PDSB PACDIB $DSUSERS/DPRTDB.txt`
export PACDIB
PACDID=`DSENV PDSB PACDID $DSUSERS/DPRTDD.txt`
export PACDID
PACDIQ=`DSENV PDSB PACDIQ $DSUSERS/DPRTDQ.txt`
export PACDIQ
PACDQI=`DSENV PDSB PACDQI $DSUSERS/DPRTQI.txt`
export PACDQI
PACDQR=`DSENV PDSB PACDQR $DSUSERS/DPRTQR.txt`
export PACDQR
PACDRQ=`DSENV PDSB PACDRQ $DSUSERS/DPRTRQ.txt`
export PACDRQ
PACDJQ=`DSENV PDSB PACDJQ $DSUSERS/DPRTJQ.txt`
export PACDJQ
```

```
PACDQJ=`DSENV PDSB PACDQJ $DSUSERS/DPRTQJ.txt`
export PACDQJ
PACDKD=`DSENV PDSB PACDKD $DSTMP/WKD`
export PACDKD
PACDKQ=`DSENV PDSB PACDKQ $DSTMP/WKQ`
export PACDKQ
PACDW1=`DSENV PDSB PACDW1 $DSTMP/W1`
export PACDW1
PACDW2=`DSENV PDSB PACDW2 $DSTMP/W2`
export PACDW2
PACDW3=`DSENV PDSB PACDW3 $DSTMP/W3`
export PACDW3
PACDW4=`DSENV PDSB PACDW4 $DSTMP/W4`
export PACDW4
DSMSG 1009 "BVPDSB$PACLANG"
rtsds BVPDSB$PACLANG
RETURN=$?
# ------------------------------------------------------------
if [ -n "$DSMSAGP" ]
then
 DSMSG 1009 "$DSMSAGP"
echo
 echo $DSMSAGP $DSUTI $NUJOB $DSUSERS $RETURN
 $DSMSAGP $DSUTI $NUJOB $DSUSERS $RETURN
fi
# ------------------------------------------------------------
case $RETURN in
 0)
   ;;
 8)
   DSMSG 1010
   DSMSG 1089
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
 *)
   DSMSG 1012 "BVPDSB$PACLANG"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 10. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DRST BATCH PROCEDURE
# * ------------------------------------------------------
# *      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.
# * -------------------------------------------------------
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DRST"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
STATUS=`dstp info $1 | grep "Server Status" | cut -d: -f2`
if [ "$STATUS" != " Inactive" -a "$STATUS" != "" ]
then
   DSMSG 1012 "DRST"
   DSMSG 1037 $1
   DSERR
   exit 12
fi
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDJ.ini
if [ -r $PACDDJ ]
then
. $DSMSDIR/config/$1/PACDDE.ini
PACDRU=`DSENV PDS380 PACDRU $DSUSERS/DRSTRU380.txt`
export PACDRU
DSMSG 1009 "BVPDS38$PACLANG"
rtsds BVPDS38$PACLANG
RETURN=$?
case $RETURN in
 0)
   ;;
 8)
   DSMSG 1012 "BVPDS38$PACLANG"
   DSMSG 1053
   DSERR
   DSRMTMP
   exit $RETURN
```

```
   ;;
 *)
   DSMSG 1012 "BVPDS38$PACLANG"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
fi
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDX.ini
. $DSMSDIR/config/$1/PACDDC.ini
. $DSMSDIR/config/$1/PACDDJ.ini
. $DSMSDIR/config/$1/PACSAVBB.ini
PACDBB=$PACSAVBB
export PACDBB
PACDMB=$DSINPUT
export PACDMB
PACDRU=`DSENV PDS400 PACDRU $DSUSERS/DRSTRU400.txt`
export PACDRU
PACDMS=`DSENV PDS400 PACDMS $DSTMP/MS`
export PACDMS
DSMSG 1009 "BVPDS400"
rtsds BVPDS400
RETURN=$?
case $RETURN in
 0)
   . $DSMSDIR/config/$1/PACDDE.ini
   . $DSMSDIR/config/$1/PACDDA.ini
   . $DSMSDIR/config/$1/PACDDX.ini
   . $DSMSDIR/config/$1/PACDDC.ini
   . $DSMSDIR/config/$1/PACSAVBJ.ini
   PACDBJ=$PACSAVBJ
   export PACDBJ

   PACDRU=`DSENV PDS450 PACDRU $DSUSERS/DRSTRU450.txt`
   export PACDRU
   PACDMS=`DSENV PDS450 PACDMS $DSTMP/MS`
   export PACDMS
   DSMSG 1009 "BVPDS450"
   rtsds BVPDS450
   RETURN=$?
   case $RETURN in
    0)
      ;;
    *)
      DSMSG 1012 "BVPDS450"
      DSMSG 1025
      DSERR
      DSRMTMP
      exit $RETURN
      ;;
```

```
   esac
   ;;
 *)
   DSMSG 1012 "BVPDS400"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 11. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DSAV BATCH PROCEDURE
# * -------------------------------------------------------
# *      VISUALAGE PACBASE-DSMS
# *
# * -------------------------------------------------------
# *            - BACKUP OF THE DSMS DATABASE -
# *
# * -------------------------------------------------------
# *
# * INPUT
# *  COL 02     : 'O'
# *  COL 03-05  : CRYPT, UNCRYPT (ENC,DEC,   )
# *
# * -------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "-------------------------------------------------------"
DSMSG 1004 "DSAV"
echo "                         =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "-------------------------------------------------------"
echo ""
DSPAUSE
STATUS=`dstp info $1 | grep "Server Status" | cut -d: -f2`
if [ "$STATUS" != " Inactive" -a "$STATUS" != "" ]
then
   DSMSG 1012 "DSAV"
   DSMSG 1037 $1
   DSERR
   exit 12
fi
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
PACDRS=`DSENV PDSBAS PACDRS $DSUSERS/DSAVRSBAS.txt`
export PACDRS
DSMSG 1009 "BVPDSBAS"
rtsds BVPDSBAS
RETURN=$?
case $RETURN in
 0)
   . $DSMSDIR/config/$1/PACDDE.ini
   . $DSMSDIR/config/$1/PACDDA.ini
   . $DSMSDIR/config/$1/PACDDX.ini
   . $DSMSDIR/config/$1/PACDDC.ini
```

```
   . $DSMSDIR/config/$1/PACSAVBB.ini
   PACDBB=$PACSAVBBNEW
   export PACDBB
   PACDMB=$DSINPUT
   export PACDMB
   PACDRU=`DSENV PDS500 PACDRU $DSUSERS/DSAVRU500.txt`
   export PACDRU
   DSMSG 1009 "BVPDS500"
   rtsds BVPDS500
   RETURN=$?
   case $RETURN in
    0)
      ;;
    *)
      DSMSG 1012 "BVPDS500"
      DSMSG 1025
      DSERR
      DSRMTMP
      exit $RETURN
      ;;
   esac
   ;;
 4)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1042
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
 *)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSMSG 1016 "BBACKUP"
sh $DSMSDIR/config/$1/BBBACKUP.ini
DSRMTMP
exit $RETURN
```

Chapter 12. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DREO BATCH PROCEDURE
# * ------------------------------------------------------
# *      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
# *
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DREO"
echo "                         =============="
```

```
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACSAVBB.ini
PACDBB=$PACSAVBB
export PACDBB
CARTE=$DSINPUT
export CARTE
PACDRH=`DSENV PDSR10 PACDRH $DSUSERS/DREPRHR10.txt`
export PACDRH
PACDRK=`DSENV PDSR10 PACDRK $DSUSERS/DREORKR10.txt`
export PACDRK
PACDW1=`DSENV PDSR10 PACDW1 $DSTMP/W1`
export PACDW1
PACDW2=`DSENV PDSR10 PACDW2 $DSTMP/W2`
export PACDW2
PACDW3=`DSENV PDSR10 PACDW3 $DSTMP/W3`
export PACDW3
DSMSG 1009 "BVPDSR10"
rtsds BVPDSR10
RETURN=$?
case $RETURN in
 0)
   PACDW2=`DSENV PDSR20 PACDW2 $DSTMP/W2`
   export PACDW2
   PACDW4=`DSENV PDSR20 PACDW4 $DSTMP/W4`
   export PACDW4
   PACDW5=`DSENV PDSR20 PACDW5 $DSTMP/W5`
   export PACDW5
   DSMSG 1009 "BVPDSR20"
   rtsds BVPDSR20
   RETURN=$?
   case $RETURN in
    0)
      PACDW3=`DSENV PDSR30 PACDW3 $DSTMP/W3`
      export PACDW3
      PACDW5=`DSENV PDSR30 PACDW5 $DSTMP/W5`
      export PACDW5
      PACDW6=`DSENV PDSR30 PACDW6 $DSTMP/W6`
      export PACDW6
      DSMSG 1009 "BVPDSR30"
      rtsds BVPDSR30
      RETURN=$?
      case $RETURN in
       0)
         . $DSMSDIR/config/$1/PACDDE.ini
         PACDBB=$PACSAVBBNEW
```

```
        export PACDBB
        PACDRR=`DSENV PDSR40 PACDRR $DSUSERS/DREORRR40.txt`
        export PACDRR
        PACDW1=`DSENV PDSR40 PACDW1 $DSTMP/W1`
        export PACDW1
        PACDW4=`DSENV PDSR40 PACDW4 $DSTMP/W4`
        export PACDW4
        PACDW6=`DSENV PDSR40 PACDW6 $DSTMP/W6`
        export PACDW6
        DSMSG 1009 "BVPDSR40"
        rtsds BVPDSR40
        RETURN=$?
        case $RETURN in
         0)
           ;;
         *)
           DSMSG 1012 "BVPDSR40"
           DSMSG 1025
           DSERR
           DSRMTMP
           exit $RETURN
           ;;
        esac
        ;;
      *)
        DSMSG 1012 "BVPDSR30"
        DSMSG 1025
  DSERR
  DSRMTMP
  exit $RETURN
        ;;
      esac
      ;;
    *)
      DSMSG 1012 "BVPDSR20"
      DSMSG 1025
      DSERR
      DSRMTMP
      exit $RETURN
      ;;
   esac
   ;;
 *)

   DSMSG 1012 "BVPDSR10"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
```

```
DSMSG 1016 "BBBACKUP"
sh $DSMSDIR/config/$1/BBBACKUP.ini
DSRMTMP
exit $RETURN
```

Chapter 13. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DEXP BATCH PROCEDURE
# * ------------------------------------------------------
# *      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
# *
# * -------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "-------------------------------------------------------"
DSMSG 1004 "DEXP"
echo "                        =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
PAC7PJ=`DSENV PDS600 PAC7PJ $DSTMP/PJ`
DSMSG 1088 "$PAC7PJ"
echo "-------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDE.ini
PACDMB=$DSINPUT
export PACDMB
PACDMV=`DSENV PDS600 PACDMV \`dirname $DSUSERS\`/MVDEXP`
export PACDMV
PAC7PJ=`DSENV PDS600 PAC7PJ $DSTMP/PJ`
export PAC7PJ
PACDRU=`DSENV PDS600 PACDMV $DSUSERS/DEXPRU600.txt`
export PACDRU
DSMSG 1009 "BVPDS600"
rtsds BVPDS600
RETURN=$?
```

```
case $RETURN in
 0)
   DSMSG 1087
   ;;
 4)
   . $DSMSDIR/config/$1/PACDDA.ini
   . $DSMSDIR/config/$1/PACDDE.ini
   PACDMV=`DSENV PDS610 PACDMV \`dirname $DSUSERS\`/MVDEXP`
   export PACDMV
   PACDRU=`DSENV PDS610 PACDRU $DSUSERS/DEXPRU610.txt`
   export PACDRU
   DSMSG 1009 "BVPDS610"
   rtsds BVPDS610
   RETURN=$?
   case $RETURN in
    0)
      ;;
    *)
      DSMSG 1012 "BVPDS610"
      DSMSG 1025
      DSERR
      DSRMTMP
      exit $RETURN
      ;;
   esac
   ;;
 *)
   DSMSG 1012 "BVPDS600"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 14. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DEXT BATCH PROCEDURE
# * -------------------------------------------------------
# *      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
# *
# * -------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "-------------------------------------------------------"
DSMSG 1004 "DEXT"
echo "                         =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "-------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDC.ini
. $DSMSDIR/config/$1/PACDDE.ini
```

```
. $DSMSDIR/config/$1/PACDDX.ini
PACDMB=$DSINPUT
export PACDMB
PACDIM=`DSENV PDSEX PACDIM \`dirname $DSUSERS\`/MVDEXT`
export PACDIM
PACDIA=`DSENV PDSEX PACDIA $DSUSERS/DEXTIAEX.txt`
export PACDIA
PACDRU=`DSENV PDSEX PACDRU $DSUSERS/DEXTRUEX.txt`
export PACDRU
PACDKQ=`DSENV PDSEX PACDKQ $DSTMP/WKQ`
export PACDKQ
PACDW0=`DSENV PDSEX PACDW0 $DSTMP/W0`
export PACDW0
PACDW1=`DSENV PDSEX PACDW1 $DSTMP/W1`
export PACDW1
PACDW2=`DSENV PDSEX PACDW2 $DSTMP/W2`
export PACDW2
PACDW3=`DSENV PDSEX PACDW3 $DSTMP/W3`
export PACDW3
PACDW4=`DSENV PDSEX PACDW4 $DSTMP/W4`
export PACDW4
PACDW5=`DSENV PDSEX PACDW5 $DSTMP/W5`
export PACDW5
PACDWI=`DSENV PDSEX PACDWI $DSTMP/WI`
export PACDWI
DSMSG 1009 "BVPDSEX$PACLANG"
rtsds BVPDSEX$PACLANG
RETURN=$?
case $RETURN in
 0)
   ;;
 *)
   DSMSG 1012 "BVPDSEX$PACLANG"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 15. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DUPT BATCH PROCEDURE
# * ------------------------------------------------------
# *      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
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DUPT"
echo "                         =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
```

```
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
PACDRS=`DSENV PDSBAS PACDRS $DSUSERS/DUPDRSBAS.txt`
export PACDRS
DSMSG 1009 "BVPDSBAS"
rtsds BVPDSBAS
RETURN=$?
case $RETURN in
 0)
   ;;
 4)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1042
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
 *)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDC.ini
. $DSMSDIR/config/$1/PACDDX.ini
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDJ.ini
PACDIM=$DSINPUT
export PACDIM
PACDRP=`DSENV PDSUP0 PACDRP $DSUSERS/DUPTRPUP0.txt`
export PACDRP
DSMSG 1009 "BVPDSUP0"
rtsds BVPDSUP0
RETURN=$?
case $RETURN in
 0)
   ;;
 *)
   DSMSG 1012 "BVPDSUP0"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 16. 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:

FIRST 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 first 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DINI BATCH PROCEDURE
# * ------------------------------------------------------
# *      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)
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DINI"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
```

```
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACSAVBB.ini
PACDBB=$PACSAVBBNEW
export PACDBB
PACDMB=$DSINPUT
export PACDMB
PACDRU=`DSENV PDSINI PACDRU $DSUSERS/DINIRUINI.txt`
export PACDRU
DSMSG 1009 "BVPDSINI"
rtsds BVPDSINI
RETURN=$?
case $RETURN in
 0)
  ;;
 *)
   DSMSG 1012 "BVPDSINI"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSMSG 1016 "BBBACKUP.ini"
sh $DSMSDIR/config/$1/BBBACKUP.ini
DSRMTMP
exit $RETURN
```

Chapter 17. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DXBJ BATCH PROCEDURE
# * -------------------------------------------------------
# *      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
# *
# * -------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "-------------------------------------------------------"
DSMSG 1004 "DXBJ"
echo "                        =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "-------------------------------------------------------"
echo ""
DSPAUSE
```

```
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACSAVBJ.ini
PACDBJ=$PACSAVBJ
export PACDBJ
PACDMB=$DSINPUT
export PACDMB
PACDIM=`DSENV PDS700 PACDIM \`dirname $DSUSERS\`/MVDXBJ`
export PACDIM
PACDRK=`DSENV PDS700 PACDRK $DSUSERS/DXBJRK700.txt`
export PACDRK
PACDSK=`DSENV PDS700 PACDSK $DSUSERS/DXBJSK700.txt`
export PACDSK
DSMSG 1009 "BVPDS700"
rtsds BVPDS700
RETURN=$?
case $RETURN in
 0)
   ;;
 *)
   DSMSG 1012 "BVPDS700"
   DSMSG 1025
   DSERR
   DSRMTP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 18. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DREN BATCH PROCEDURE
# * ------------------------------------------------------
# *      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
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DREN"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDX.ini
. $DSMSDIR/config/$1/PACSAVBB.ini
PACDBB=$PACSAVBB
export PACDBB
. $DSMSDIR/config/$1/PACSAVBJ.ini
PACDBJ=$PACSAVBJ
export PACDBJ
PACDMB=$DSINPUT
```

```
export PACDMB
PACDW0=`DSENV PDSMS PACDW0 $DSTMP/W0`
export PACDW0
PACDW1=`DSENV PDSMS PACDW1 $DSTMP/W1`
export PACDW1
PACDW2=`DSENV PDSMS PACDW2 $DSTMP/W2`
export PACDW2
PACDB3=$PACSAVBBNEW
export PACDB3
PACDJB=$PACSAVBJNEW
export PACDJB
PACDIA=`DSENV PDSMS PACDIA $DSUSERS/DRENIAMS.txt`
export PACDIA
PACDIK=`DSENV PDSMS PACDIK $DSUSERS/DRENIKMS.txt`
export PACDIK
PACDJK=`DSENV PDSMS PACDJK $DSUSERS/DRENJKMS.txt`
export PACDJK
PACDIS=`DSENV PDSMS PACDIS $DSUSERS/DRENISMS.txt`
export PACDIS
PACDKK=`DSENV PDSMS PACDKK $DSUSERS/DRENKKMS.txt`
export PACDKK
PACDLK=`DSENV PDSMS PACDLK $DSUSERS/DRENLKMS.txt`
export PACDLK
DSMSG 1009 "BVPDSMS$PACLANG"
rtsds BVPDSMS$PACLANG
RETURN=$?
case $RETURN in
0)
 ;;
8)
 DSMSG 1012 "BVPDSMS$PACLANG"
 DSMSG 1090
 DSERR
 DSRMTMP
 exit $RETURN
 ;;
10)
 DSMSG 1012 "BVPDSMS$PACLANG"
 DSMSG 1091
 DSERR
 DSRMTMP
 exit $RETURN
 ;;
12)
 DSMSG 1012 "BVPDSMS$PACLANG"
 DSMSG 1092
 DSERR
 DSRMTMP
 exit $RETURN
 ;;
*)
 DSMSG 1012 "BVPDSMS$PACLANG"
 DSMSG 1025
 DSERR
 DSRMTMP
```

```
 exit $RETURN
 ;;
esac
# ************************************************************
DSMSG 1010
if [ -f "$PACSAVBJNEW" ]
then
   DSMSG 1016 "BJBACKUP"
   . $DSMSDIR/config/$1/BJBACKUP.ini
fi
if [ -f "$PACSAVBBNEW" ]
then
   DSMSG 1016 "BBBACKUP"
   . $DSMSDIR/config/$1/BBBACKUP.ini
   DSMSG 1093 "BB"
fi
DSRMTMP
exit $RETURN
```

Chapter 19. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DPDF BATCH PROCEDURE
# * ------------------------------------------------------
# *      VISUALAGE PACBASE-DSMS
# *
# * ------------------------------------------------------
# *     - ACCESS FACILITY PRE-PROCESSING -
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DPDF"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
DAF80=`DSENV DAFD10 DAF80 \`dirname $DSUSERS\`/DAF`
export DAF80
COB80=`DSENV DAFD10 COB80 \`dirname $DSUSERS\`/COB`
export COB80
DAFREP=`DSENV DAFD10 DAFREP $DSUSERS/DAFREP.txt`
export DAFREP
DSMSG 1009 "BVDAFD10"
rtsds BVDAFD10
RETURN=$?
case $RETURN in
0)
 ;;
```

```
*)
 DSMSG 1012 "BVDAFD10"
 DSMSG 1025
 DSERR
 DSRMTMP
 exit $RETURN
 ;;
esac
# ************************************************************
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 20. 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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DUPD BATCH PROCEDURE
# * ------------------------------------------------------
# *      VISUALAGE PACBASE-DSMS
# *
# * ------------------------------------------------------
# *         - BATCH UPDATE FROM DAF TABLES -
# *
# * ------------------------------------------------------
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------"
DSMSG 1004 "DUPD"
echo "                        =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDA.ini
PACDRS=`DSENV PDSBAS PACDRS $DSUSERS/DUPDRSBAS.txt`
```

```
export PACDRS
DSMSG 1009 "BVPDSBAS"
rtsds BVPDSBAS
RETURN=$?
case $RETURN in
 0)
   ;;
 4)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1042
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
 *)
   DSMSG 1012 "BVPDSBAS"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
# ************************************************************
PACDGY=$DSINPUT
export PACDGY
PACDIM=`DSENV PDS900 PACDIM $DSTMP/IM`
export PACDIM
DSMSG 1009 "BVPDS900"
rtsds BVPDS900
RETURN=$?
case $RETURN in
 0)
   ;;
 *)
   DSMSG 1009 "BVPDS900"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
# ************************************************************
. $DSMSDIR/config/$1/PACDDA.ini
. $DSMSDIR/config/$1/PACDDC.ini
. $DSMSDIR/config/$1/PACDDX.ini
. $DSMSDIR/config/$1/PACDDE.ini
. $DSMSDIR/config/$1/PACDDJ.ini
PACDIM=`DSENV PDSUP0 PACDIM $DSTMP/IM`
export PACDIM
PACDRP=`DSENV PDSUP0 PACDIM $DSUSERS/DUPDRPUP0.txt`
export PACDRP
DSMSG 1009 "BVPDSUP0"
rtsds BVPDSUP0
RETURN=$?
case $RETURN in
```

```
 0)
   ;;
 *)
   DSMSG 1012 "BVPDSUP0"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSRMTMP
exit $RETURN
```

Chapter 21. 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 (refer to next subchapter).

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

```
#!/bin/sh
#@(#)DSMS xxx xxx (R) DLVB BATCH PROCEDURE
# * -------------------------------------------------------
# *      VISUALAGE PACBASE-DSMS
# *
# * -------------------------------------------------------
# *   - CHANGE LOW VALUE CHARACTERS INTO BLANK CHARACTERS -
# *
# *   OPTION : SUBMIT PROCEDURE WITH PARM='DATA'
# *            TO PROCESS DATA OPTION
# * -------------------------------------------------------
```

```
# *
# Parameter control
. $DSMSDIR/system/proc/DSINIT.ini
echo ""
echo "------------------------------------------------------------"
DSMSG 1004 "DLVB"
echo "                          =============="
DSMSG 1047 "$DSBASE"
DSMSG 1005 "$DSMSDIR/config/$1"
DSMSG 1006 "$DSTMP"
DSMSG 1073 "$DSUSERS"
DSMSG 1007 "$DSINPUT"
echo "------------------------------------------------------------"
echo ""
DSPAUSE
DSMKDIR
# ************************************************************
. $DSMSDIR/config/$1/PACSAVBB.ini
PACDBB=$PACSAVBB
export PACDBB
PACDB1=$PACSAVBBNEW
export PACDB1
DSMSG 1009 "BVPDSLVB"
rtsds BVPDSLVB
RETURN=$?
case $RETURN in
 0)
   ;;
 *)
   DSMSG 1012 "BVPDSLVB"
   DSMSG 1025
   DSERR
   DSRMTMP
   exit $RETURN
   ;;
esac
DSMSG 1010
DSMSG 1016 "BBBACKUP.ini"
sh $DSMSDIR/config/$1/BBBACKUP.ini
DSRMTMP
exit $RETURN
```

Part Number: DEDIX000351A - 7639

Printed in USA