English

Fujitsu Software BS2000

UDS/SQL V2.9
Database Operation

User Guide

Valid for:
UDS/SQL V2.9C

December 2025

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000.info@fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Copyright © 2025 Fujitsu

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Table of Contents

Database Operation 9
1 Preface 10
1.1 Structure of the UDS/SQL documentation 11
1.2 Objectives and target groups of this manual 16
1.3 Summary of contents 17
1.4 Changes since the last edition of the manuals 18
1.5 Notational conventions 20
1.5.1 Warnings and notes 21
1.5.2 Non-SDF notational conventions 22
1.5.3 SDF syntax representation 23
2 The Database Handler (DBH) 28
2.1 How the independent DBH works 32
2.2 How the linked-in DBH works 39
2.3 The session 42
2.3.1 Starting DBH 43
2.3.2 Defining and altering the DB configuration 59
2.3.3 Terminating the DBH 61
2.4 Parallel database operation 64
3 DBH load parameters 65
4 Administration 134
4.1 Administration of UDS/SQL via UDSADM 135
4.1.1 UDSADM statements 136
4.2 Administration of UDS/SQL via DCAM 144
4.3 Administration of UDS/SQL via INFORM-PROGRAM 147
4.4 The Database Administrator Language DAL 148
4.4.1 Rolling back one or all transactions (ABORT) 156
4.4.2 Handling access locks at database and realm level (ACCESS) 157
4.4.3 Activating online extensibility (ACT) 159
4.4.4 Attaching databases, realms and passwords (ADD) 163
4.4.5 Adding new entries to the distribution table (&ADD DISTRIBUTION) 168
4.4.6 Displaying the number of processed base interface blocks (%BIB) 170
4.4.7 Assigning a configuration to a different host (&CHANGE DISTRIBUTION) . . 171
4.4.8 Writing checkpoints (CHECKPOINT) 172
4.4.9 Terminating the session or administration (CLOSE) 174
4.4.10 Terminating UDS-D operation (&CLOSE DISTRIBUTION) 175
4.4.11 Terminating a transaction in the PTC state (COMMIT) 176
4.4.12 Continuing DISPLAY SQL output (CONTINUE) 177

4.4.13 Deactivating online extensibility (DEACT) 178
4.4.14 Listing information on the DB configuration (DISPLAY) 180
4.4.15 Displaying all or part of the distribution table (&DISPLAY DISTRIBUTION) . 226
4.4.16 Displaying information on an SQL conversation (DISPLAY SQL) 229
4.4.17 Displaying the number of messages to the UDS/SQL DBH (%DML) 236
4.4.18 Detaching databases, realms and passwords (DROP) 237
4.4.19 Deleting entries from the distribution table (&DROP DISTRIBUTION) 241
4.4.20 Generating a memory dump (DUMP) 243
4.4.21 Generating a memory dump for the linked-in DBH (%DUMP) 244
4.4.22 Executing an online DBTT extension (EXTEND DBTT) 245
4.4.23 Executing an online realm extension (EXTEND REALM) 247
4.4.24 Releasing SQL resources (FORGET SQL) 248
4.4.25 Canceling the STOP state of transactions (GO) 249
4.4.26 Locking entries in the distribution table (&LOCK DISTRIBUTION) 250
4.4.27 Changing the settings of DEFAULT-SUPPORT and RESERVE-SUPPORT of the ALOG files (MODIFY ALOG) 252
4.4.28 Changing the storage allocation of the ALOG files (MODIFY ALOG-SIZE) . 253
4.4.29 Changing the allocation of volumes for the RLOG files (MODIFY LOG) . . . 254
4.4.30 Changing the amount of storage for RLOG files (MODIFY LOG-SIZE) 256
4.4.31 Modifying PTCSYNCH values (MODIFY PTCSYNCH) 257
4.4.32 Changing reserve volumes for the RLOG files (MODIFY RESERVE) 258
4.4.33 Check and note a new UDS/SQL pubset declaration (NEW PUBSETS) . . . 259
4.4.34 Selecting new RLOG files (NEW RLOG) 262
4.4.35 Performing pending requests (PERFORM) 263
4.4.36 Assigning and changing a password (&PWD DISTRIBUTION) 265
4.4.37 Reactivating online extensibility for a Realm (REACT INCR) 266
4.4.38 Canceling pending requests (RESET ORDERS) 267
4.4.39 Saving the distribution table (&SAVE DISTRIBUTION) 268
4.4.40 Starting UDS-D operation (&START DISTRIBUTION) 269
4.4.41 Stopping the execution of transactions (STOP) 270
4.4.42 Terminating secondary subtransactions in the PTC state (&SYNCHRONIZE DISTRIBUTION) 271
4.4.43 Aborting the session immediately (%TERM) 272
4.4.44 Removing locks from distribution table entries (&UNLOCK DISTRIBUTION) . . 273
5 High availability 275
5.1 Uninterrupted database operation 276
5.2 Interaction with the transaction monitor openUTM 277
5.3 Formatting the databases 278
5.4 Error tolerance 279
5.5 Distributed processing 280

5.6 Media recovery 281
5.7 Restart functions 282
5.8 Using standard BS2000 procedures to increase data security 283
5.9 Protection against failure of local resources 284
5.10 Using UDS/SQL utility routines 285
6 Resource extension and reorganization during live operation 286
6.1 System requirements for online extension 287
6.1.1 Allocating storage space 288
6.1.2 DMS-related aspects of online realm extension 289
6.1.3 Estimating the extensibility of a realm 290
6.1.4 Effect of utility routines on the online extensibility of realms 291
6.1.5 Database layout version and realm layout version 292
6.1.6 Database page formatting during realm extension 293
6.1.7 Free place administration during realm extension 294
6.2 Online realm extension 295
6.2.1 The online realm extension process 296
6.2.2 Troubleshooting 297
6.2.3 Possible responses after an unsuccessful online realm extension 298
6.2.4 DAL commands and utility routines for online realm extension 299
6.2.5 Activating online extensibility when creating databases 301
6.3 Online DBTT extension 302
6.3.1 Activating online DBTT extension 303
6.3.2 Execution of an online DBTT extension 304
6.3.3 Initiating an explicit online DBTT extension 305
6.3.4 Response in error situations 306
6.3.5 DAL commands and utility routines for online DBTT extension 307
6.4 Online reorganization with the UDS online utility 308
6.5 Automatic realm extension by means of utility routines 309
7 Saving and recovering a database in the event of errors 311
7.1 The UDS/SQL recovery concept 312
7.2 Transaction recovery 313
7.2.1 Error types and their recovery 314
7.2.1.1 Errors in the application program 315
7.2.1.2 Errors in the system 316
7.2.2 Handling the required files 317
7.2.2.1 RLOG file 318
7.2.2.2 Session log file (SLF) 320
7.2.2.3 Status file 321
7.3 Media recovery 322
7.3.1 Making a database backup 323
7.3.1.1 Consistency points 324

7.3.1.2 Saving a consistent database . . . 325
7.3.1.3 Saving a database online . . . 326
7.3.1.4 Shadow database . . . 327
7.3.2 Maintaining ALOG files . . . 328
7.3.3 Reconstructing a database . . . 330
7.4 Selecting a recovery concept . . . 332
7.5 Examples . . . 335
8 Optimizing performance . . . 350
8.1 Optimizing I/O behavior . . . 351
8.2 Optimizing usage of working memory with the subsystem functionality of UDS /SQL . . . 353
8.3 Optimizing processor utilization with the independent DBH . . . 355
8.3.1 Load parameters that affect processor utilization . . . 356
8.3.2 Performance-oriented BS2000 settings . . . 358
8.3.3 Optimized task communication . . . 359
8.3.4 Interpreting communication counters of the UDS/SQL monitor . . . 360
8.3.5 Example of a peak-load configuration . . . 361
9 Using BS2000 functionality . . . 363
9.1 Using pubsets in UDS/SQL . . . 364
9.2 Using job variables in UDS/SQL . . . 367
9.2.1 Pubset declaration job variable . . . 368
9.2.1.1 Use of the UDS/SQL pubset declaration in the DBH . . . 371
9.2.1.2 Use of the UDS/SQL pubset declaration in the utility routines . . . 372
9.2.1.3 Other uses of the UDS/SQL pubset declaration . . . 373
9.2.2 Session job variable . . . 374
9.2.3 Database job variable . . . 378
9.2.4 Information on using the job variables . . . 383
9.2.5 BMEND job variable . . . 384
9.3 Using multiple UDS/SQL versions concurrently . . . 385
9.4 Using DAB caching for UDS/SQL . . . 389
9.5 Using data mirroring for UDS/SQL . . . 390
9.6 Using HSMS in UDS/SQL . . . 391
9.7 Using FASTPAM in UDS/SQL . . . 392
9.8 Using the full address space in UDS/SQL . . . 393
9.9 Determining time via the GET-TIME subsystem . . . 394
9.10 Assigning passwords to UDS/SQL files . . . 395
9.11 Accounting in UDS/SQL openUTM operation . . . 397
9.12 UDS/SQL users and user groups . . . 398
9.12.1 Access control via openUTM . . . 399
9.12.2 Defining UDS/SQL user groups and granting access privileges . . . 401

9.13 ACS (Alias Catalog Service) 404
9.14 Unicode concept in UDS/SQL 405
10 The SQL conversation 407
11 Output of database operating and status values with UDSMON 408
11.1 Description of functions 410
11.2 System environment 412
11.3 Statements and commands for UDSMON 414
11.4 DISPLAY output of the UDS monitor to intermediate files 419
11.5 Command sequence for starting and operating UDSMON 421
11.6 Example 422
11.7 Description of UDS/SQL monitor output to data display terminal 423
11.8 Description of UDS/SQL monitor output to printer 438
11.9 Description of UDS/SQL monitor output to a file 439
11.10 Transfer of monitor counters to openSM2 449
12 General functions of the utility routines 451
12.1 Starting utility routines 452
12.2 Assigning the database name 453
12.3 Checking the UDS/SQL pubset declaration 454
12.4 Automatic realm extension 455
12.5 Outputting database information in a neutral format 456
12.5.1 Differences between CSV output and output to SYSLST 457
12.5.2 Controlling the CSV output with DISPLAY statements 458
12.5.3 CSV output in BOUTLOAD utility 459
12.5.4 Examples of usage 460
12.6 Job switches 461
13 Using IQS 462
13.1 Files, modules and procedures used by IQS 463
13.2 Diagnostic aids 465
14 Using UDS-D 466
14.1 Brief product description 467
14.2 Notes on using UDS-D 469
14.2.1 Design and programming 470
14.2.2 Reorganization 471
14.3 Working of UDS-D 472
14.3.1 The subtransaction concept 473
14.3.2 Primary and secondary subtransactions 474
14.3.3 Multi-DB programs for distributed processing 475
14.3.4 Communication between configurations 476
14.4 Ensuring cross-configuration consistency with a two-phase commit 478
14.4.1 Sequential flow of a two-phase commit for distribution via UDS-D 479

14.4.2 Sequential flow of a two-phase commit in UDS-D for distribution via UDS-D and openUTM 481
14.4.3 PTC state 483
14.4.4 Terminating the PTC state 487
14.5 Database operation with UDS-D 488
14.5.1 The DBH when using UDS-D 489
14.5.2 Configuration-based password protection 492
14.5.3 The distribution table 493
14.5.3.1 Structure of the input file for the distribution table 494
14.5.3.2 Reading the input file 496
14.5.3.3 Structure of the distribution table 497
14.5.3.4 Overview of DAL commands for distribution tables 498
14.5.4 Time-controlled monitoring of connections and transactions 499
14.5.4.1 Monitoring the logical connection to remote application programs 500
14.5.4.2 Monitoring secondary subtransactions that are not in the PTC state . . . 501
14.5.4.3 Monitoring secondary subtransactions in the PTC state 502
14.5.4.4 Monitoring primary subtransactions that were terminated by the database administrator 503
14.5.5 Starting and terminating UDS-D mode 504
14.5.5.1 Starting UDS-D mode 505
14.5.5.2 Terminating UDS-D mode 506
14.5.5.3 Restarting UDS-D mode 507
14.5.5.4 Examples for UDS-D mode 508
14.5.6 Effects of the PTC state 516
14.5.7 Administering multiple configurations 521
14.5.7.1 Local administration 522
14.5.7.2 Central administration 523
14.5.8 Altering the DB configuration: notes for UDS-D 525
14.5.9 Using the UDS/SQL configuration on a virtual host 526
14.6 DBH load parameters for UDS-D 528
14.7 The Database Administrator Language DAL for UDS-D 530
15 Appendix 532
15.1 Function codes of the DML statements 533
16 Glossary 536
17 Abbreviations 574
18 Related publications 577

Database Operation

1 Preface

The **U**niversal **D**atabase **S**ystem UDS/SQL is a high-performance database system based on the structural concept of CODASYL. Its capabilities, however, go far beyond those of CODASYL as it also offers the features of the relational model. Both models can be used in coexistence with each other on the same data resources.

COBOL DML, CALL DML and (ISO standard) SQL are available for querying and updating data. COBOL DML statements are integrated in the COBOL language; SQL statements can be used in DRIVE programs.

To ensure confidentiality, integrity and availability, UDS/SQL provides effective but flexible protection mechanisms that control access to the database. These mechanisms are compatible with the openUTM transaction monitor.

The data security concept provided by UDS/SQL effectively protects data against corruption and loss. This concept combines UDS/SQL-specific mechanisms such as logging updated information with BS2000 functions such as DRV (Dual Recording by Volume).

If the add-on product UDS-D is used, it is also possible to process data resources in BS2000 computer networks. UDS/SQL ensures that the data remains consistent throughout the network. Distributed transaction processing in both BS2000 computer networks and networks of BS2000 and other operating systems can be implemented using UDS/SQL together with openUTM-D or openUTM (Unix/Linux/Windows). UDS/SQL can also be used as the database in client-server solutions.

The architecture of UDS/SQL (e.g. multitasking, multithreading, DB cache) and its structuring flexibility provide a very high level of throughput.

1.1 Structure of the UDS/SQL documentation

The “Guide through the manuals” section explains which manuals and which parts of the manuals contain the information required by the user. A glossary gives brief definitions of the technical terms used in the text.
In addition to using the table of contents, users can find answers to their queries either via the index or by referring to the running headers.

Guide through the manuals

The UDS/SQL database system is documented in five manuals:

- UDS/SQL Design and Definition
- UDS/SQL Application Programming
- UDS/SQL Creation and Restructuring
- UDS/SQL Database Operation
- UDS/SQL Recovery, Information and Reorganization

Further manuals describing additional UDS/SQL products and functions are listed in section "Additional Manuals" of Table 1 .

For a basic introduction the user should refer to chapters 2 and 3 of the “Design and Definition” manual; these chapters describe

- reasons for using databases
- the CODASYL database model
- the relational database model with regard to SQL
- the difference between the models
- the coexistence of the two database models in a UDS/SQL database
- the characteristic features of UDS/SQL

How the manuals are used depends on the user’s previous knowledge and tasks. Table 1 serves as a guide to help users find their way through the manuals.

Examples

A user whose task it is to write COBOL DML programs should look up the column “COBOL/CALL DML Programming” under “User task” in the second line of table 1. There, the following chapters of the “Design and Definition” manual are recommended:

General information	B = Basic information
Schema DDL	D = Detailed information
SSL	D = Detailed information
Subschema DDL	L = Learning the functions

In the same column the user can also see which chapters of the other manual are of use.

Database administrators who are in charge of database administration and operation will find the appropriate information under the column “Administration and Operation”.

Contents of the five main manuals	User task							
	Design and definition	COBOL/ CALL DML programming	SQL programming	Creation and restructuring	Administration and operation	Working with openUTM	Working with IQS	Working with UDS-D
Manual UDS/SQL Design and Definition								
Preface	B	–	–	–	–	B	B	–
General information	B	B	B	B	B	B	–	–
Designing the database	B	–	–	–	–	–	–	–
Schema DDL	L	D	–	L	L	–	–	–
SSL	L	D	–	L	L	–	–	–
Subschema DDL	L	L	–	L	L	–	–	–
Relational schema	L	–	D	–	–	–	–	–
Structure of pages	D	–	–	D	D	–	–	–
Structure of records and tables	D	–	–	D	D	–	–	–
Reference section	S	–	–	S	–	–	–	–
Manual UDS/SQL Application Programming								
Preface	–	B	–	–	–	B	B	–
Overview	–	B	–	–	–	–	–	–
Transaction concept	–	L	–	L	L	D	D	–
Currency table	–	L	–	L	L	–	–	–
DML functions	D	L	–	L	–	–	–	–
Using DML	–	L	–	D	–	–	–	–
COBOL DML reference section	–	L	–	–	–	–	–	–
CALL DML reference section	–	L	–	–	–	–	–	–
Testing DML functions using DMLTEST	–	L	–	–	–	–	–	–

Manual UDS/SQL Creation and Restructuring								
Preface	–	–	–	B	–	B	B	–
Overview	–	–	–	B	B	–	–	–
Database creation	–	–	–	L	–	–	–	–
Defining access rights	–	–	–	L	–	–	–	–
Storing and unloading data	D	–	–	L	–	D	–	–
Restructuring the database	D	–	–	L	–	–	–	–
Renaming database objects	D	–	–	L	–	–	–	–
Database conversion	D	–	–	L	–	–	–	–
Database conversion using BTRANS24	–	–	–	D	–	–	–	–

Manual UDS/SQL Database Operation								
Preface	–	–	–	–	B	B	B	–
The database handler	–	–	–	–	L	–	–	D
DBH load parameters	–	–	–	–	L	–	–	D
Administration	–	–	–	–	L	–	–	D
High availability	–	–	–	–	B	–	–	–
Resource extension and reorganisation during live operation	D	–	–	–	B	–	–	–
Saving and recovering a database in the event of errors	D	–	–	D	L	D	–	D
Optimizing performance	–	–	–	–	D	–	–	D
Using BS2000 functionality	–	–	–	–	D	–	–	–
The SQL conversation	–	–	–	–	L	–	–	–
UDSMON	–	–	–	–	D	–	–	–
General functions of the utility routines	–	–	–	–	D	–	–	–
Using IQS	–	–	–	L	D	–	D	–
Using UDS-D	D	D	–	D	D	D	–	D
Function codes of DML statements	–	D	–	–	D	–	–	–
Manual UDS/SQL Recovery, Information and Reorganization								
Preface	–	–	–	–	B	B	B	–
Updating and reconstructing a database	D	–	–	D	L	D	–	–
Checking the consistency of a database	–	–	–	–	L	–	–	–
Output of database information	D	–	–	D	L	–	–	–
Executing online services	D	–	–	D	L	–	–	–
Database reorganization	D	–	–	D	L	–	–	–
Controlling the reuse of deallocated database keys	D	–	–	D	L	–	–	–

Additional Manuals								
UDS/SQL Messages	D	D	D	D	D	D	D	D
UDS/SQL System Reference Guide	S	S	–	S	S	S	S	S
IQS	–	–	–	D	D	–	L	–
ADILOS	–	–	–	–	D	–	L	–
KDBS	–	L [1]	–	D	–	–	–	–
SQL for UDS/SQL Language Reference Manual	–	–	D	–	D	–	–	–

Table 1: Guide through the manuals

[1] only for COBOL-DML

B	provides basic information for users with no experience of UDS/SQL
L	helps the user learn functions
D	provides detailed information
S	provides a reference to syntax rules for practical work with UDS/SQL

Additional notes on the manuals

References to other manuals appear in abbreviated form. For example:

(see the “Application Programming” manual, CONNECT)

advises the user to look up CONNECT in the “Application Programming” manual. The complete titles of the manuals can be found under “Related publications“ at the back of the manual.

UDS/SQL Messages

This manual contains all messages output by UDS/SQL. The messages are sorted in ascending numerical order, or in alphabetical order for some utility routines.

UDS/SQL System Reference Guide

The UDS/SQL System Reference Guide gives an overview of the UDS/SQL functions and formats.

SQL for UDS/SQL Language Reference Manual

This manual describes the SQL DML language elements of UDS/SQL.
In addition to UDS/SQL-specific extensions, the language elements described include dynamic SQL as an essential extension of the SQL standard.

1.2 Objectives and target groups of this manual

This manual is intended for database administrators whose job it is to plan and monitor database operation.

Database administrators must be able to perform all the steps involved in designing a database (including the setting up of schema, subschema and SSL) and know how to write DB application programs.

They must also have an in-depth knowledge of BS2000, be familiar with the UDS/SQL transaction concept and have a basic knowledge of the files of a UDS/SQL database and the UDS/SQL utility routines.

1.3 Summary of contents

What does this manual contain?

This manual describes administration and operating activities necessary for trouble-free database operation:

- database operation using the Database Handler
- starting, administering and terminating the session
- administration via UDSADM, DCAM and SEND-MSG
- database backup and recovery
- optimizing performance
- using BS2000 functionality for UDS/SQL
- the SQL conversation
- outputting database operating and status values using the UDS/SQL monitor UDSMON
- general functions of the utility routines
- using IQS
- using UDS-D

Readme file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at https://bs2manuals.ts.fujitsu.com. You will also find the Readme files on the Softbook DVD.

Information under BS2000

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

`SYSRME.<product>.<version>.<lang>`

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `/SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at https://bs2manuals.ts.fujitsu.com.

1.4 Changes since the last edition of the manuals

The main changes introduced in UDS/SQL V2.9 in comparison with Version V2.8 are listed in table 2 below together with the manuals and the sections in which the changes are described. If a specific topic has been dealt with in more than one manual, the manual in which a detailed description appears is listed first. The following codes are used in the “Manual” column for the individual manuals involved:

DES	Design and Definition	DBO	Database Operation
APP	Application Programming	RIR	Recovery, Information and Reorganization
CRE	Creation and Restructuring	MSG	Messages

Topic	**Manual**	**Chapter**
Changes in V2.9A		
FIND-/FETCH-7 with DESCENDING KEY: Suspension of the restriction		
The restriction for DESCENDING KEY is omitted	APP	7
Record references in COBOL programs:		
The new DDL-statement GENERATE-REC-REF generates a data field and condition names for the access to record references	CRE	3
Changing settings of ALOG files while the database is in use		
New DAL command DISPLAY ALOG shows ALOG settings	DBO	4
New DAL commands MODIFY ALOG/MODIFY ALOG-RES and MODIFY-ALOG-SIZE change ALOG settings	DBO	4
Change of the restrictions for the UDS Online Utility		
WAIT-FOR-TRANSACTION offers the possibility to wait until the locked source page is released by the locking transaction	RIR	8
With SET-RELOCATE-PARAMETERS a behaviour for the case that pages are locked can be specified also for *INDEX-LEVEL-TABLE-PAGES (CLASH-HANDLING)	RIR	8
BRENAME with after-image logging: The function “Renaming database objects (BRENAME, BALTER)” can also be executed when After Image Logging is activated. Thus logging gaps can be avoided		
New behaviour in a renaming cycle	CRE	7
After a renaming process a data base update can be executed	RIR	2
Specifying the size for DBTT extensions		
New operand EXT of the DAL command ACT DBTT-INCR	DBO	4

The BSTATUS output additionally contains the value of the EXT operand	RIR	6
New data type FIXED REAL BINARY 63		
Expansion of the syntax representation	DES	4, 9
Changes in messages to consider the new data type	MSG APP	2, 4 10
Changes in V2.9B		
BINILOAD of CSV files		
Explanation how BINILOAD handles CSV files	CRE	2, 5
Changes to consider new feature	MSG	3
BINILOAD with records of variable length		
Biniload can work with records with variable length	CRE	5
Changes to consider new feature	MSG	3
Default values for new fields in BALTER		
New BALTER statement to fill fields with specified values	CRE	6
Changes to consider new feature	MSG	3
Number of extents in DB-JV		
Number of extents of the ALOG file are added in Database Job Variable	DBO	9

Table 2: Changes in version V2.9 compared to V2.8

1.5 Notational conventions

This section provides an explanation of the symbols used for warnings and notes as well as the notational conventions used to describe syntax rules.

1.5.1 Warnings and notes

i	Points out particularly important information
! **CAUTION!**	Warnings

1.5.2 Non-SDF notational conventions

Language element	Explanation	Example
KEYWORD	Keywords are shown in underlined uppercase letters. You must specify at least the underlined parts of a keyword.	DATABASE-KEY MANUAL
OPTIONAL WORD	Optional words are shown in uppercase letters without underlining. Such words may be omitted without altering the meaning of a statement.	NAME IS ALLOWED PAGES
variable	Variables are shown in italic lowercase letters. In a format which contains variables, a current value must be entered in place of each variable.	*item-name* *literal-3* *integer*
{ either | or }	Exactly one of the expressions enclosed in braces must be specified. Indented lines belong to the preceding expression. The braces themselves must not be specified.	{ CALC | INDEX } { VALUE IS | VALUES ARE }
[optional]	The expression in square brackets can be omitted. UDS /SQL then uses the default value The brackets themselves must no be specified.	[IS *integer*] [WITHIN *realm-name*]
... or ,...	The immediately preceding expression can be repeated several times if required. The two language elements distinguish between repetitions which use blanks and those which use commas.	*item-name*,... {SEARCH KEY.....}...
..... or . .	Indicates where entries have been omitted for reasons of clarity. When the formats are used, these omissions are not allowed.	SEARCH KEY IS ..... RECORD NAME .. .
.	The period must be specified and must be followed by at least one blank. The underline must not be specified.	SET SECTION. 03 *item-name*..... .
Space	Means that at least one blank has to be specified.	USING CALC

Table 3: Notational conventions

All other characters such as ( ) , . ; “ = are not metacharacters;
they must be specified exactly as they appear in the formats.

1.5.3 SDF syntax representation

This syntax description is based on SDF Version 4.7. The syntax of the SDF command/statement language is explained in the following three tables.

Table 4 **: Metasyntax**

Certain characters and representations are used in the statement formats; their meaning is explained in table 4.

Table 5 **: Data types**

Variable operand values are represented in SDF by data types. Each data type represents a specific value set. The number of data types is limited to those described in table 5.

The description of the data types is valid for all commands and statements. Therefore only deviations from table 5 are explained in the relevant operand descriptions.

Table 6 **: Data type suffixes**

The description of the “integer” data type in table 6 also contains a number of items in italics. The italics are not part of the syntax, but are used merely to make the table easier to read.

The description of the data type suffixes is valid for all commands and statements. Therefore only deviations from table 6 are explained in the relevant operand descriptions.

Representation	**Meaning**	**Examples**
UPPERCASE LETTERS	Uppercase letters denote keywords. Some keywords begin with *.	OPEN DATABASE COPY-NAME = *NONE
=	The equal sign connects an operand name with the associated operand values.	CONFIGURATION-NAME = <name 1..8>
< >	Angle brackets denote variables whose range of values is described by data types and their suffixes (Tables 5 and 6).	DATABASE = <dbname>
Underscoring	Underscoring denotes the default value of an operand.	SCHEMA-NAME = *STD
/	A slash separates alternative operand values.	CMD = *ALL / <dal-cmd>
(...)	Parentheses denote operand values that initiate a structure.	*KSET-FORMAT(...)
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	USER-GROUP-NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) | HOST = <host>

| | |	A vertical bar identifies related operands within structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical preceding an operand corresponds to the depth of the structure.	USER-GROUP-NAME = *ALL-EXCEPT(...) *ALL-EXCEPT(...) | NAME = *KSET-FORMAT(...) | *KSET-FORMAT(...) | | HOST = <host> | | ...
,	A comma precedes further operands at the same structure level.	,SPACE = STD
list-poss(n):	`list-poss` signifies that the operand values following it may be entered as a list. If a value is specified for (n), the list may contain no more than that number of elements. A list of two or more elements must be enclosed in parentheses.	NAME = list-poss(30): <subschema-name>

Table 4: Metasyntax

Data type	Character set	Special rules
alog-seq-no	0..9	1..9 characters
appl	A..Z 0..9 $,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or $, #, @ Strings of less than 8 characters are filled internally with underscore characters.
catid	A..Z 0..9	1..4 characters must not start with the string PUB
copyname	A..Z 0..9	1..7 characters, starting with A..Z
c-string	EBCDIC characters	1..4 characters Must be enclosed in single quotes; the letter C may be used as a prefix. Single quotes within c-string must be specified twice.

csv-filename	A..Z 0..9 Structure identifier: hyphen	1..30 characters Must be enclosed in single quotes
dal-cmd	A..Z 0..9 hyphen	1..64 characters
date	0..9 Structure identifier: hyphen	Date specification Input format: yyyy-mm-dd yyyy : year; may be 2 or 4 digits long mm : month dd : day
dbname	A..Z 0..9	1..17 characters, starting with A..Z
device	A..Z 0..9 $,#,@ Structure identifier: hyphen	5..8 characters, starting with A..Z or 0..9 String that can consist of a number of substrings separated by hyphens and and whicch corresponds to a device. In the dialog guidance, SDF shows the permissible operand values. Information as the possible devices can be found in the relevant operand description.
host	A..Z 0..9 $,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or $, #, @ Strings of less than 8 characters are filled internally with underscore characters.
integer	0..9,+,-	+ or - may only be the first character.
kset	A..Z 0..9 $,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or $, #, @ Strings of less than 8 characters are filled internally with underscore characters.
name	A..Z 0..9 $,#,@	1..8 characters Must not consist only of 0..9 and must not start with a digit
realm-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that may consist of a number of substrings by hyphens; first character: A..Z

realmref	0..9	1..3 characters
record-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z In the case of record types with a search key it is recommendable to use names with no more than 26 characters, otherwise the set name created implicitly (SYS_...) will be truncated in accordance with the restriction on the name length for sets.
recordref	0..9	1..3 characters
schema-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z
set-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z
structured-name	A...Z 0...9 $, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A...Z or $, #, @
subschema-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z
time	0..9 Structure identifier: colon	Time-of-day specification `{ hh:mm:ss | hh:mm | hh }` hh, mm, ss: Leading zeros may be omitted
userid	A..Z 0..9 $,#,@	1..8 characters, beginning with A..Z or $,#,@ BPRIVACY: Strings of less than 8 characters are filled internally with underscore characters.
volume	A..Z 0..9 $,#,@	1..6 characters starting with A..Z or 0..9

x-string	Hexadecimal: 00..FF	1..8 characters Must be enclosed in single quotes and prefixed with the letter X.There may be an odd number of characters

Table 5: Data types

Suffix	**Meaning**
x..y *unit*	For the “integer” data type: range specification. x Minimum value permitted for “integer”. x is an (optionally signed) integer. y Maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* for “integer” only: additional units. The following units may be specified: *Mbyte, Kbyte, seconds*

Table 6: Data type suffixes

2 The Database Handler (DBH)

A UDS/SQL database cannot be accessed by the database application program directly, unlike the data of a conventional file for example. Instead, access to the database is controlled by the Database Handler (DBH), which thus relieves the application program of all administrative duties, e.g. such as arise when data is stored.

The Database Handler exists in two forms:

- independent DBH
- linked-in DBH.

The **independent DBH** is implemented as an autonomous task family. It accepts database calls from the application programs, performs accesses to the database and transports the data to be processed between the database and the work area in the application program.

The DBH also monitors all database activities and conducts a dialog with the database administrator, i.e. it reports any errors that occur and receives the database administrator’s instructions.

The independent DBH has multi-database capability, i.e. can manage several databases simultaneously. All databases in the multi-DB configuration can be accessed by a database application program via this DBH, even within a single transaction.

The ONLINE-PRIVACY utility routine is used with the independent DBH.The UDS-ONLINE-UTILITY utility routine can be used with the independent DBH and with the linked-in DBH.

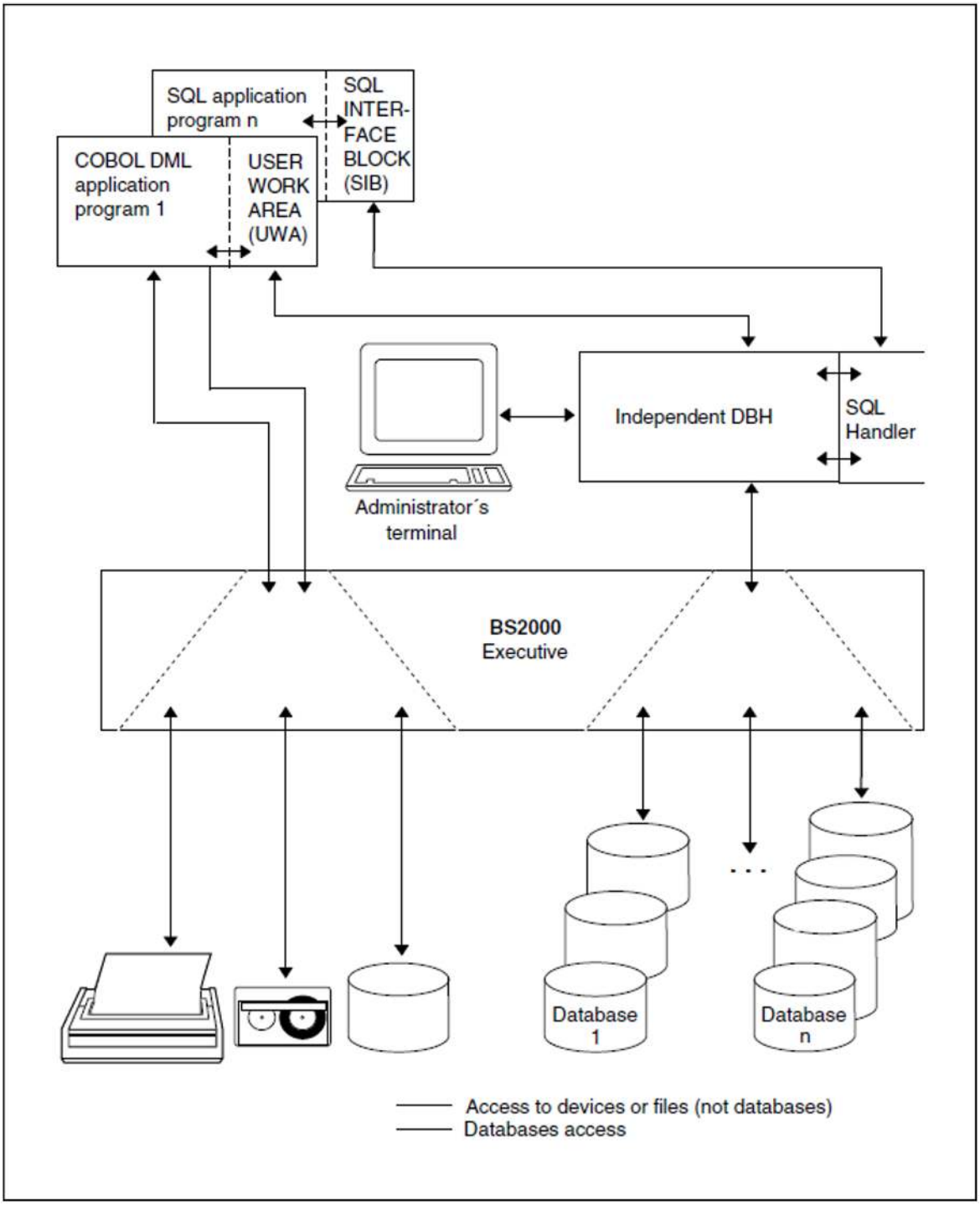


Figure 1: Principle of the independent DBH

The **linked-in DBH** is not an autonomous program. The modules of the linked-in DBH are linked into the database application program; they run as part of this program. When the program wants to access a database, it invokes the DBH. The DBH accesses the database and writes data from the UWA of the program to the database or delivers data requested from the database to the UWA.

As the task communication which is necessary between the tasks of the task family and the user tasks when the independent DBH is used does not apply in the case of the linked-in DBH, the runtime of an application program which is used individually can be shortened.

> The linked-in DBH cannot process SQL statements.

The linked-in DBH has multi-DB capability, i.e. a DB application program that works with the linked-in DBH can access several databases. Unless the DBH opens these databases for read-only access (SHARED-RETRIEVAL), they are barred to all other programs during the linked-in application.
UDS/SQL utility routines that need the DBH in general operate with the linked-in DBH. This also applies to the DDL and SSL compilers, which work with a mono-DB configuration, i.e. always address only a single database.
The UDS-ONLINE-UTILITY utility routine can be used with the linked-in DBH and also with the independent DBH.

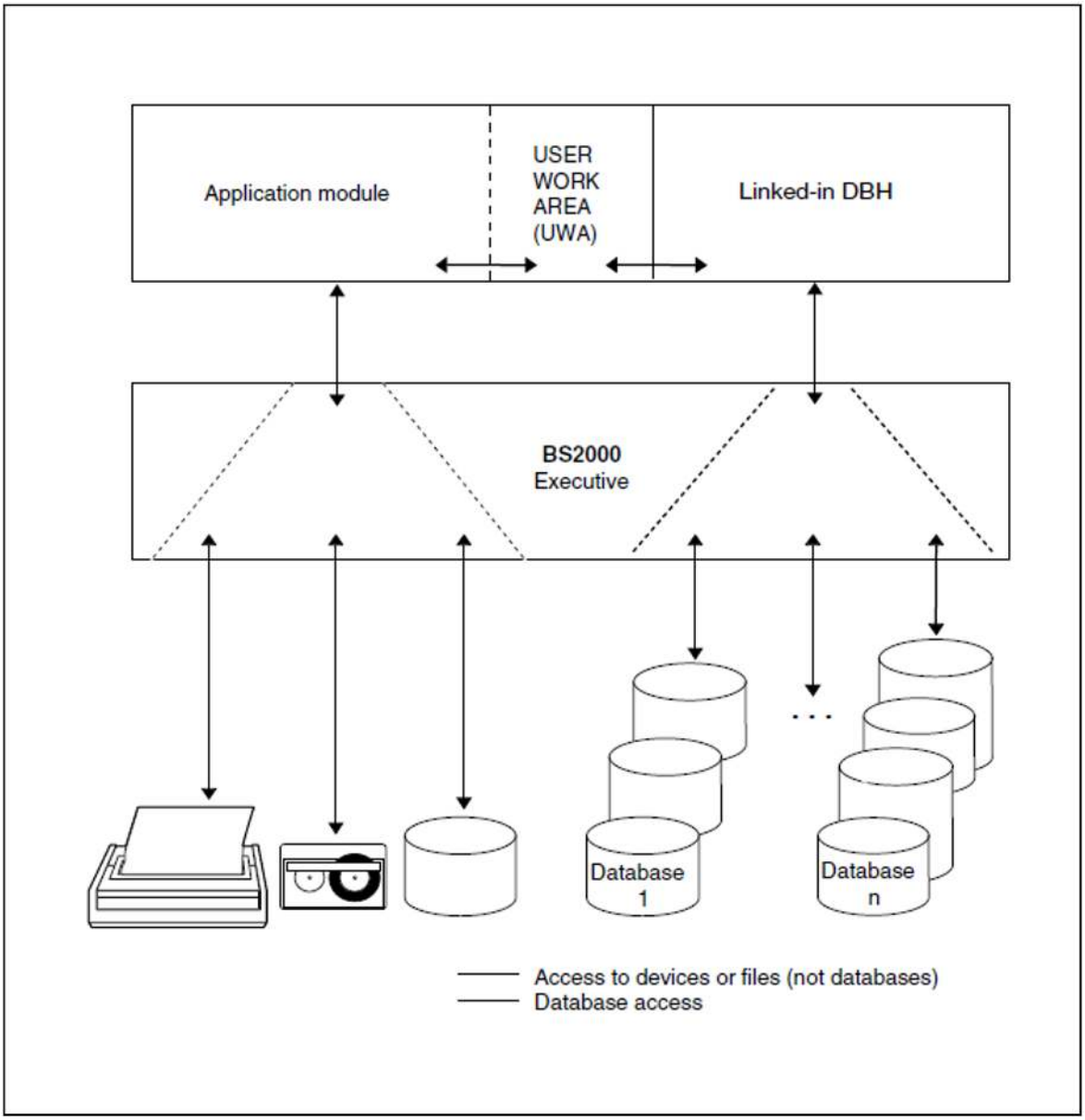


Figure 2: Principle of the linked-in DBH

2.1 How the independent DBH works

The independent DBH consists of a number of tasks (also called UDS/SQL tasks below):

- the master task
- one or more server tasks
  (whose number is defined by the load parameter PP SERVERTASK)
- the administrator task

The **master task** (module UDSSQL) prepares the actual session phase when the DBH is started and performs all the necessary tests for this; it also controls DBH termination.

The master task loads the server task(s) as one or more ENTER jobs on starting the DBH and subsequently terminates them on ending the DBH.

In addition, the master task must perform special functions during the session in order to guarantee that everything runs smoothly.

To be on the safe side you are advised to start the master task in an ENTER job.It is also possible to start the master task interactively. In this case you *must* make sure that it is never interrupted unnecessarily and that you always enter RESUME-PROG after issuing BS2000 commands.

The **server task** (module UDSSUB) accepts the request from the communication pool and executes it. It accesses the database via the common pool and transfers data which the application program wishes to write into or read from the database from the communication pool to the common pool, and vice versa. When the server task has finished processing the call, it informs the application program.

The standard inputs and outputs from the server task are processed asynchronously (load parameter PP IO=ASYNC). It is therefore sufficient to start one server task on a monoprocessor system and a maximum of one server task per processor on a multiprocessor system (see section “Optimizing processor utilization with the independent DBH”). These server tasks must then have a correspondingly high priority (see ”RUN-PRIO“ in chapter "Starting DBH" and especially section “Performance-oriented BS2000 settings”).

The database administrator uses the **administrator task** (module UDSADM) to control the session. Use of the administrator task ensures that the master task can run without interruption and that the database administrator is connected to the session in an easy way (see section “Administration of UDS/SQL via UDSADM”). This form of administration should generally be used.

An area of the **communication pool** is reserved for each transaction. The following information is stored in this area when a call is processed:

- the transaction identifier
- the call and its parameters
- the data to be transferred

The **common pool** is a memory area that can be accessed by all UDS/SQL tasks of the DBH. It contains, among other things:

- system tables
- the system buffer pools and exclusive buffer pools for database pages

The system buffer pools contain a number of 2, 4, or 8-Kbyte buffers, each of which can accommodate one database page. The size of the system buffer pool in Mbytes is determined by the load parameters PP 2KB-BUFFER-SIZE, PP 4KB-BUFFER-SIZE or PP 8KB-BUFFER-SIZE.

For each database, an exclusive buffer pool can be created in addition to the system buffer pools. The size of the exclusive buffer pool can be defined with the load parameter PP DBNAME or the DAL command ADD DB.

Each database page to be processed by a server task is read into a buffer of the corresponding system buffer pool or exclusive buffer pool and remains there until

- the session is terminated or
- the database is detached with DROP DB or
- the buffer is required for another database page.

The server task thus does not need to access the database for each DML call, but can operate in the system buffer pools or exclusive buffer pools.

When SQL is used, there is a UDS/SQL conversation for every SQL application (see chapter “The SQL conversation”), and there are system tables relating to these conversations in the communication pool and in the common pool.

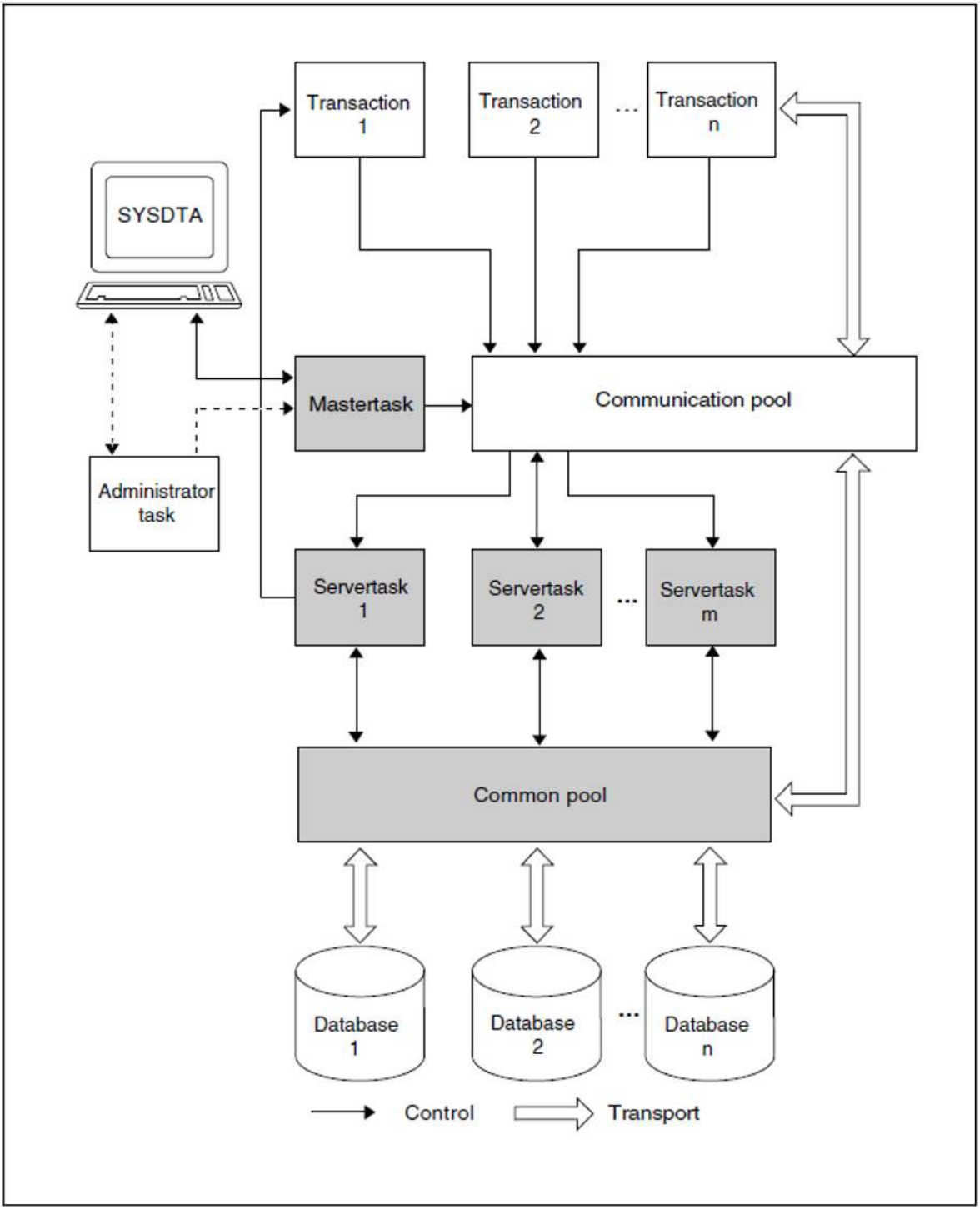


Figure 3: The independent DBH

Processing of a COBOL DML statement by the independent DBH

The following paragraphs outline the processing of a DML statement by the independent DBH using a COBOL program as an example; the following diagram illustrates the processing sequence:

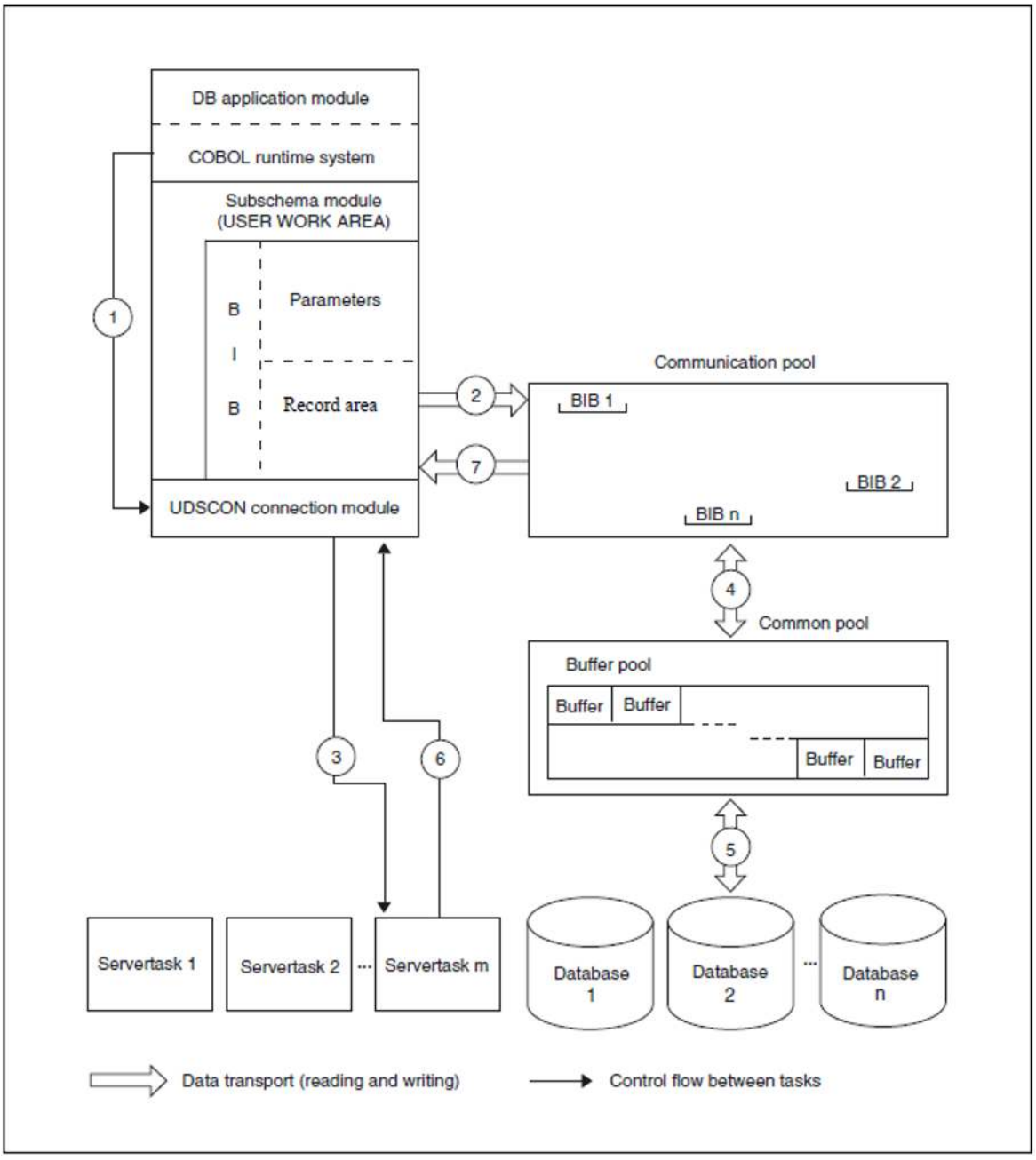


Figure 4: Handling of a COBOL DML statement by the independent DBH

1. A work area, the record area, is available to the application program through the subschema module. This area is at the same time part of the BASE INTERFACE BLOCK (BIB), which is the interface between UDS/SQL and the database user.The record area of the BIB is used to pass the data associated with a DML statement.The COBOL runtime system places the required parameters for the DML statement into the BIB parameter area. In a CALL statement, the COBOL runtime system passes the address of the BIB to the connection module.

2. The connection module transfers the BIB to the communication pool, in which an area is reserved for this purpose at the start of a transaction.
3. This BIB is passed to a server task and processed.

   If no server task is available, the BIB must wait until a server task becomes free. The connection module places itself in wait status until the DML statement has been processed.
4. In order to process the DML statement, the server task accesses the BIB in the communication pool and the buffer pools in the common pool.
   If the DML statement is to read data from the database, the server task transfers the data in the buffer pools to the appropriate BIB in the communication pool. Conversely, if the DML statement is to write data to the database, the server task transfers the data in the BIB to the buffer pools.
   However, the server task does not write a buffer from the buffer pools back to the database until another database page needs the space in the buffer.
5. If the database page into which the DML statement is to write the data of the BIB or from which it is to read data is not in the buffer pools, a search is made in the specified database for the appropriate database page, which is then read into the buffer pools.
6. After the DML statement has been processed, the server task activates the user task associated with the transaction, thereby removing the connection module from wait status.
7. The connection module writes the processed BIB from the communication pool back to the application program. The connection module returns control to the application program. The application program can now process the data in the record area.

Processing of a CALL DML statement by the independent DBH

The following paragraphs describe the processing of a DML statement using a CALL DML program as an example; the following diagram illustrates the processing sequence:

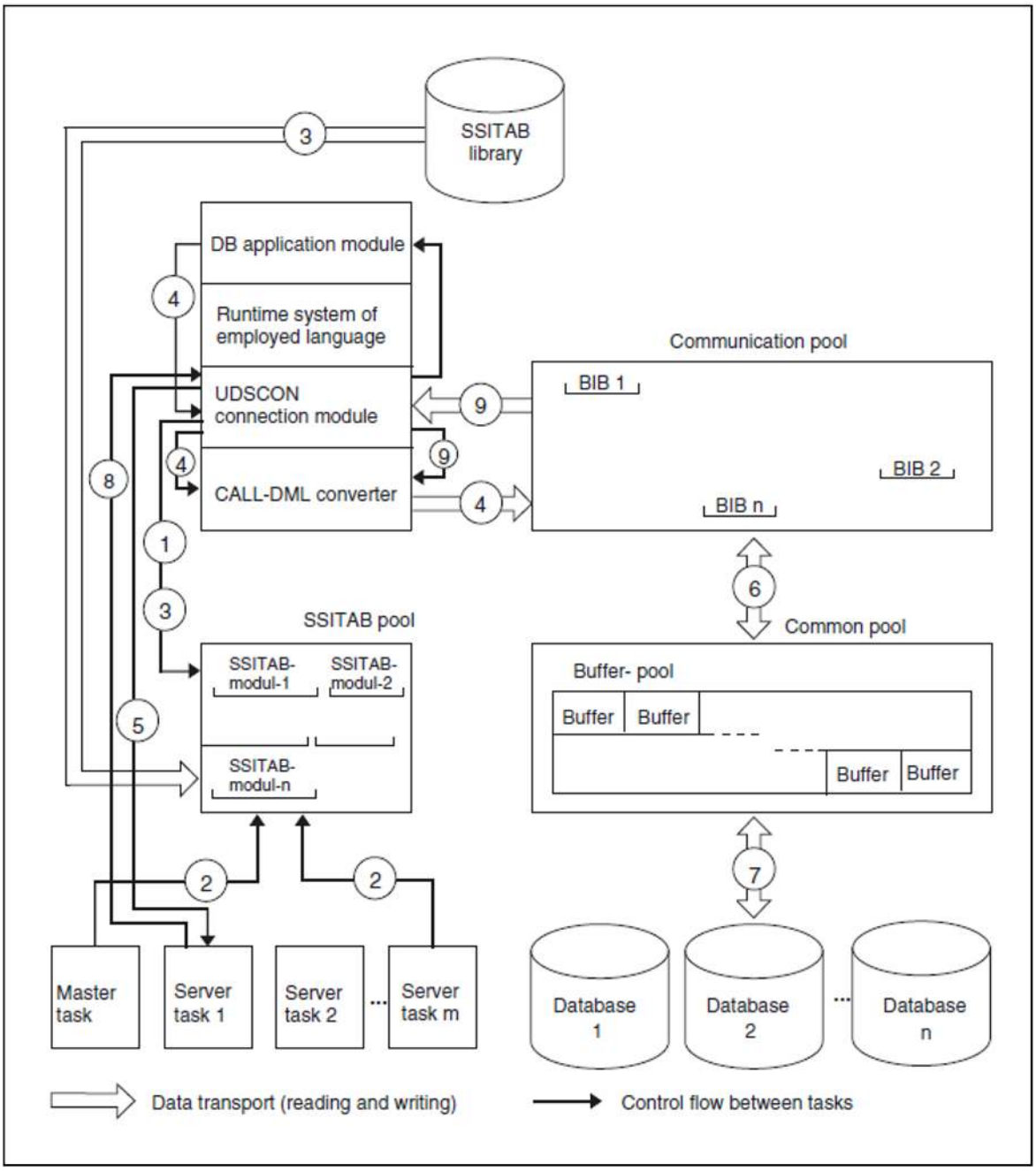


Figure 5: Handling of a CALL DML statement independent DBH

The role of the SSITAB pool and the connection module UDSCON is explained here.

1. If the application program is the first CALL DML application program of the current UDS/SQL session, the SSITAB pool is set up by the connection module.

2. All tasks of the UDS/SQL task family must also attach themselves to the SSITAB pool at this time. If this is accomplished, it will be possible to work with the CALL DML from now until the end of the session. The SSITAB pool remains in existence until the end of the session.
3. Each application program that works with the CALL DML attaches itself to the pool and checks whether the SSITAB module is already in the SSITAB pool. If this is not the case, an attempt is made to load the SSITAB module dynamically from a library which was assigned with the link name $UDSSSI before the application program was started. For detailed information on the dynamic loading of SSITAB modules, see "Starting DBH" and the “Application Programming” manual, section “Linking, loading and starting a UDS/SQL-TIAM application program”.
4. Now if the application module issues a DML request at the CALL interface, the request is received by the connection module. The connection module forwards the request to the CALL DML converter, which converts the request into a BIB that UDS/SQL can understand. The connection module then writes the BIB into the communication pool.
5. This BIB is passed to a server task and processed. If no server task is available, the BIB must wait until a server task becomes free. The connection module places itself in wait status until the DML statement has been processed. Only after this does the application program regain control.
6. In order to process the DML statement, the server task accesses the BIB in the communication pool and the buffer pools in the common pool.
   If the DML statement is to read data from the database, the server task transfers the data in the buffer pools to the appropriate BIB in the communication pool. Conversely, if the DML statement is to write data to the database, the server task transfers the data in the BIB to the buffer pools.
   Generally, however, the server task does not write a buffer from the buffer pools back to the database until another database page needs the space in the buffer.
7. If the database page into which the DML statement is to write the data of the BIB or from which it is to read data is not in the buffer pools, a search is made in the specified database for the appropriate database page, which is then read into the buffer pools.
8. After the DML statement has been processed, the server task activates the user task associated with the transaction, thereby removing the connection module from wait status.
9. When the BIB has been processed by UDS/SQL, the connection module takes it and passes it on to the CALL DML converter, which converts the results from the BIB to the CALL interface that the user can understand.

2.2 How the linked-in DBH works

The linked-in DBH is not an autonomous program; its modules are linked into the database application program and run as part of that program.

The linked-in DBH has multi-DB capability, i.e. an application program working with the linked-in DBH can access several databases simultaneously. The program reserves these databases exclusively, which means that, unless the DBH opens them for read-only access (SHARED-RETRIEVAL), no other program can work with them at the same time.

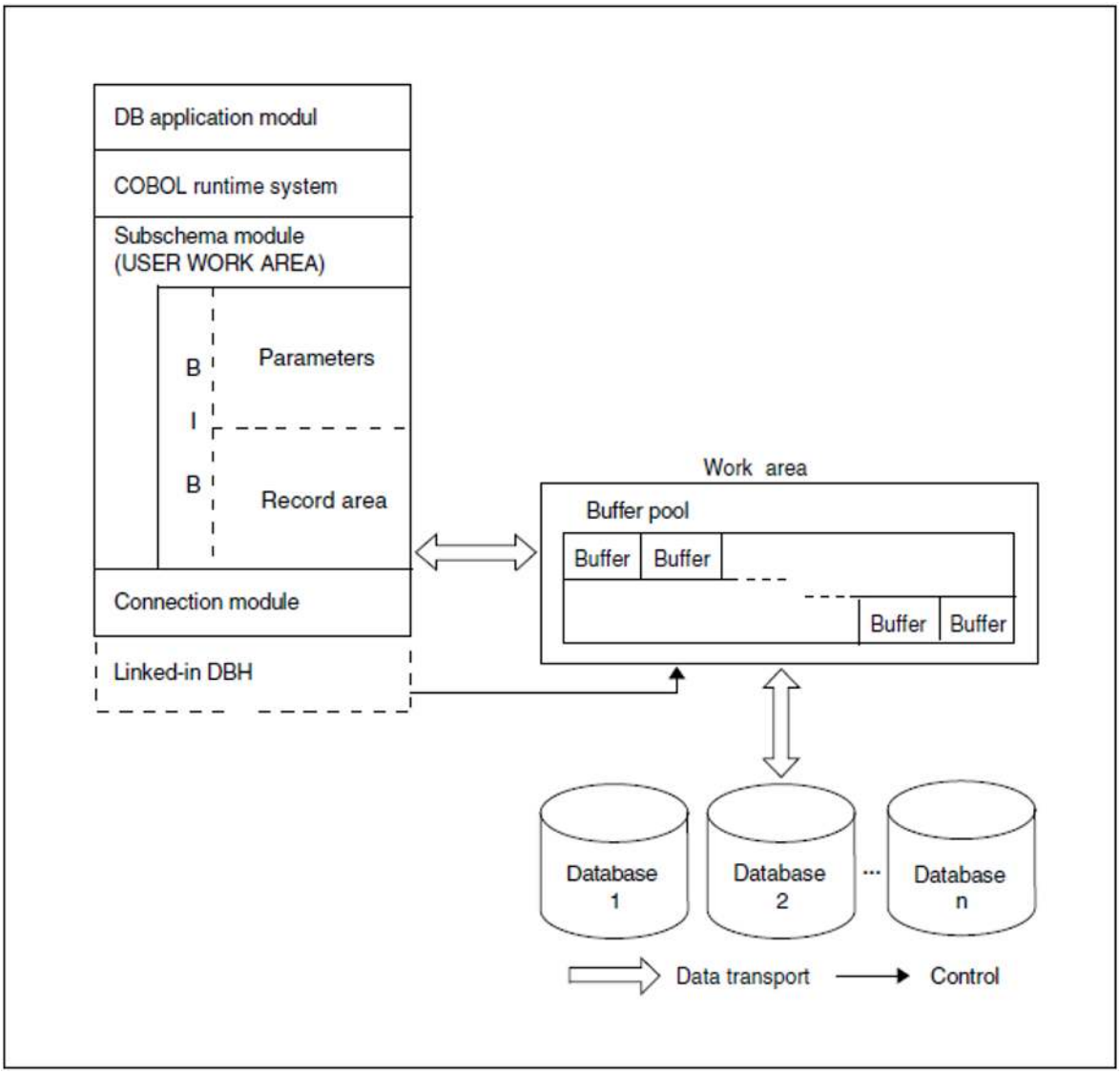


Figure 6: The linked-in DBH

A special connection module must be linked into the application program in addition to the subschema modules (UDSLNKL or UDSLNKA, see the “Application Programming” manual, section “Linking, loading and starting a UDS /SQL-TIAM application program”). This ensures that when the application is started the dynamic binder loader DBL loads the modules of the linked-in DBH. As in the case of the independent DBH, no processing is done directly against the database. Instead, all work is done in the buffer pools of the DBH work area, which corresponds to the buffer pools in the common pool of the independent DBH.
In the case of the linked-in DBH, however, this work area is not located in a memory area shared by a number of tasks (common memory), but in the memory area of the associated task.

The following example shows how a COBOL program can use the linked-in DBH to process a DML statement. The processing sequence is illustrated in the following diagram:

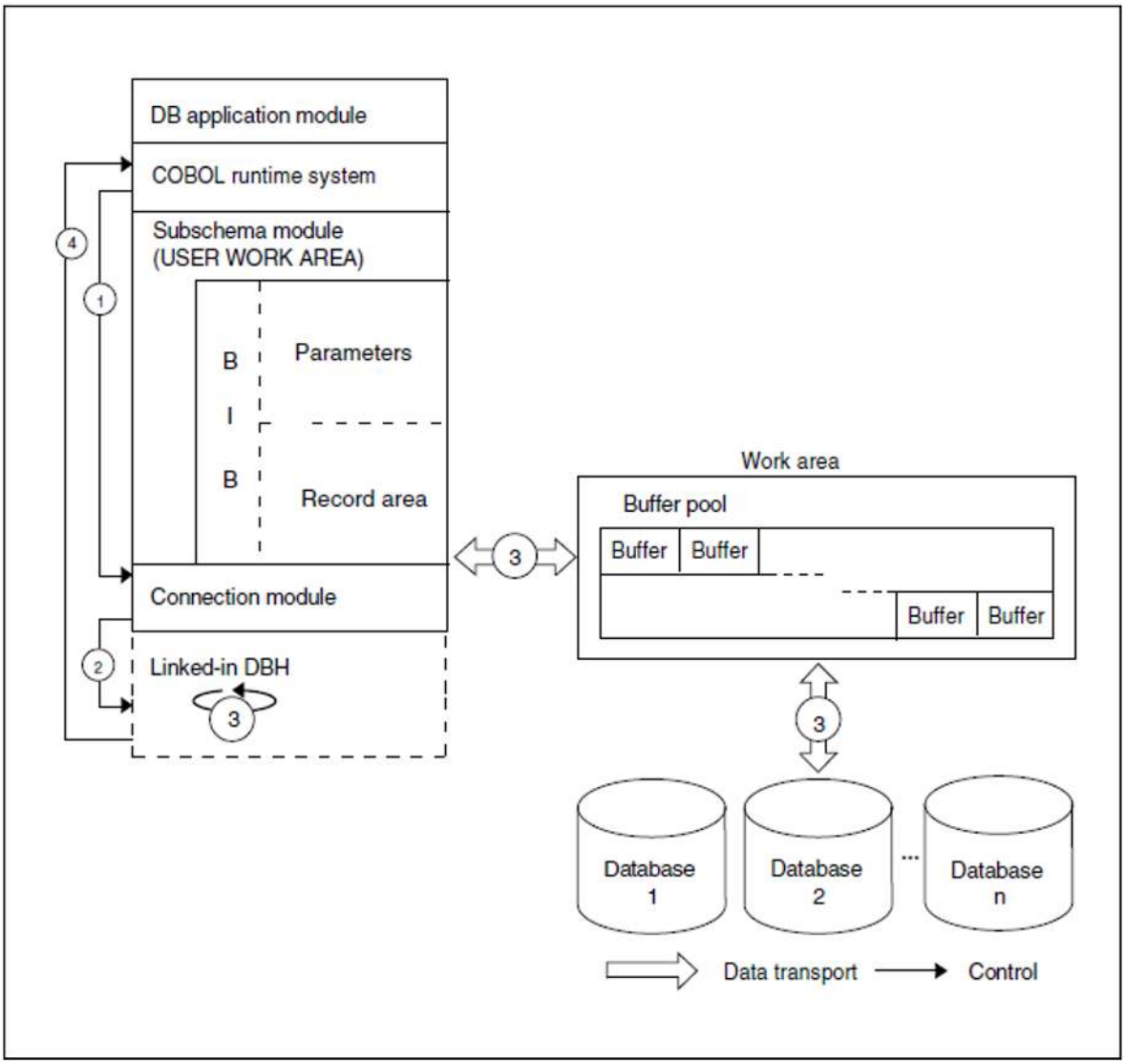


Figure 7: Processing of COBOL DML statement by the linked-in DBH

1. The subschema module contains the record area, which is at the same time part of the BASE INTERFACE BLOCK (BIB). The record area can be addressed by the application program via the SUB-SCHEMA SECTION.

   The DB application module transfers the data associated with the relevant DML statement to the record area of the BIB.

   The COBOL runtime system places the information needed to process the DML call into the parameter area of the BIB and passes the address of the BIB to the connection module.
2. The connection module calls the processing modules of the linked-in DBH and passes the address of the BIB.
3. The DBH processes the DML call, i.e.:
   - If the DML statement is to read data from the database, the DBH supplies the BIB with the required data. To this end, the DBH accesses the buffer pools and conducts a search for the relevant database page. If the page is not yet in the buffer pools, the DBH reads it in from the database.

- If the DML statement is to write data to the database, the DBH writes the data from the BIB to the relevant database page in the buffer pools. If the page is not yet contained in the buffer pools, the DBH reads it in from the database.

4. When the DBH finishes processing the DML call, it returns control to the application program.

2.3 The session

A session is the period of time between the activation of the DBH (i.e. a session start) and its **normal** termination (session end). During the session, there may be an interrupt (session abort), in which case the “session restart” is regarded as a continuation of a session.

The databases which are attached to a DBH in the course of a session form a DB configuration.

The DB configuration for the session is defined either with DBH load parameters (see chapter “DBH load parameters”) or, after the DBH has been started, with Database Administrator Language commands(see section “The Database Administrator Language DAL”).

A DB configuration consisting of several databases is referred to as a multi-DB configuration; a configuration consisting of one database only is referred to as a mono-DB configuration.

During a session with the **independent DBH** it is possible to use DAL commands to change the DB configuration, i. e. databases can be attached or detached, whether or not they are consistent. If an inconsistent database is attached it is made consistent by means of a warm start.

Altering the DB configuration after a session abort is known as reconfiguration (see "Reconfiguration" in chapter "Defining and altering the DB configuration" and section “Error types and their recovery”).

2.3.1 Starting DBH

Independent DBH

You start the independent DBH by loading the master task with the START-UDS-DBH command (for further start options, see point 11 in chapter "Starting DBH") and terminate it with the DAL command CLOSE. During the session you can intervene to control the workflow using commands from the database administrator language.

You can enter the DAL commands via the administration interface (see section “Administration of UDS/SQL via UDSADM” and section “Administration of UDS/SQL via DCAM”).

Syntax of the START-UDS-DBH command:

START-UDS-DBH

VERSION = ***STD /** <product-version>

,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>

,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767 *seconds*>

,**RESID**ENT**-PAGE**S = [***PAR**AMETERS](...)

[***PAR**AMETERS](...)

| **MIN**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

| ,**MAX**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

VERSION =
Product version of the DBH which is to be started.

VERSION = *STD
No explicit specification of the product version. The product version is selected as follows:

1. The version predefined with the /SELECT-PRODUCT-VERSION command.
2. The highest DBH version installed with IMON.

VERSION = <product-version>
Explicit specification of the product version in the form mm.n[a[kk]].

You are recommended always to specify the version in full, e.g. 02.9C00, in order to facilitate migration in the event of correction packages.

MONJV =
Specifies a monitor job variable to monitor the DBH run.

MONJV = *NONE
No monitor job variable is used.

MONJV = <filename 1..54 without-gen-vers>
Name of the job variable to be used.

During the program run the system sets the job variable to the following values:

$R	Program running
$T	Program successfully terminated
$A	Program terminated with error

CPU-LIMIT =
Maximum CPU time in seconds which the DBH may take to execute.

CPU-LIMIT = *JOB-REST
The remaining CPU time for the BS2000 job is to be used for the task.

CPU-LIMIT = <integer 1..32767 *seconds*>
Only the time specified should be used.

RESIDENT-PAGES = *PARAMETERS(...)
Number of resident memory pages which are required for the DBH run.
This operand must be specified if pages are to be made resident in the program by means of a CSTAT macro (see the “Executive Macros” manual). The permissible number of resident memory pages can be influenced by the operator.
If the operand is missing (this corresponds to MIN=*STD, MAX=*STD), the memory requests are taken from the first record in the program; the file must be open to do this.

MINIMUM = *STD / <integer 0..32767 *4Kbyte*>
Minimum number of resident memory pages required.

MAXIMUM = *STD / <integer 0..32767 *4Kbyte*>
Maximum number of resident memory pages required.

Linked-in DBH

The linked-in DBH is started by loading and starting the application program which incorporates the linked-in DBH. The linked-in DBH is started automatically:

when COBOL-DML is used
at the start of the DB application program

when CALL DML is used exclusively
with the first READYC call

when KDBS is used
with the first OPDB call

when IQS is used
at the start of the IQS load module C.L.IQS.

The DBH is closed on termination of the application program.

Commands for starting DBH

The following command overview is based on the standard case in which the DBH of a UDS/SQL version which was installed using IMON and is managed centrally in the SCI (Software Configuration Inventory) is used. The sections headed “Private installation” list commands which are not relevant for the standard case but only if you are using a privately installed UDS/SQL version which is not managed in the SCI.

No.	Commands	independent DBH	Linked-in DBH
1.	/SET-FILE-LINK LINK-NAME=DATABASE ,FILE-NAME=*configuration-name*	X	X
2.	[/CREATE-FILE FILE-NAME=*confname*.TEMPO.*nnn* ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=*device*[,SPACE=...])]	X	X
3.	[/CREATE-FILE FILE-NAME=*confname*.SLF ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=*device*[,SPACE=...])]	X	X
4.	/CREATE-FILE FILE-NAME=*confname*.DBSTAT ,SUPPRESS-ERRORS=*FILE-EXISTING /CREATE-FILE FILE-NAME=*confname*.DBSTAT.SAVE ,SUPPRESS-ERRORS=*FILE-EXISTING	X	X
5.	[/MODIFY-JOB-SWITCHES OFF=28]		X
6.	[/ASSIGN-SYSDTA TO={*LIBRARY-ELEMENT(...) | *SYSCMD}]	X	X
	[/ADD-FILE-LINK LINK-NAME=PPFILE ,FILE-NAME=*filename*]	X	X
7.	[/SET-FILE-LINK LINK-NAME=UDSDBHJV ,FILE-NAME=*jvname*]	X	X
8.	[/SET-JV-LINK LINK-NAME=UDSPS01 ,JV-NAME=*jvname*]	X	X
9.	[/ADD-FILE-LINK LINK-NAME=$UDSSSI ,FILE-NAME=*SSITAB-modlib*]		X
	[/ADD-FILE-LINK LINK-NAME=$UDSPLEX ,FILE-NAME=*PLITAB-modlib*]		X
10.	[/ADD-FILE-LINK LINK-NAME=$UDSKONF ,FILE-NAME=*UDSTRTAB-modlib*]	X	X

11.	/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL ,VERSION=*nn.nann*,SCOPE=*TASK	X	X
	[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D ,VERSION=*nn.nann*,SCOPE=*TASK]	X	
	Private installation (additional commands)		
	[/ADD-FILE-LINK LINKNAME=$UDSLIB ,FILE-NAME=*UDS/SQL-modlib*]	X	X
	[/ADD-FILE-LINK LINKNAME=$UDSDLIB ,FILE-NAME=*UDS-D-modlib*]	X	
	[/ADD-FILE-LINK LINKNAME=$UDSKLIB ,FILE-NAME=*KDBS-modlib*]		X
12.	/START-UDS-DBH [RESIDENT-PAGES=*PAR(MIN=...,MAX=...)]	X	
	Private installation (alternative command)		
	/START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE (LIB=*UDS/SQL-modlib* ,ELEM=UDSSQL)	X	
13.	/START-PROGRAM FROM-FILE=(LIB=*lib* ,ELEM=*elem*,RUN-MODE=*ADVANCED [(NAME-COLLISION=*STD)],PROG-MOD=*ANY) [,RESIDENT-PAGES=*PAR(MIN=..., MAX=...)]		X

Table 7: Command for startting DBH

1. This SET-FILE-LINK command assigns the name of the DB configuration to the session.

   configuration-name

   Freely selectable name for the database configuration, which may be identical with *dbname* in mono-DB mode. The following rules apply:

   - *configuration-name* must not exceed 41 characters in length;
   - every *configuration-name* used in a BS2000 session must be unique in the first seven characters;
   - *configuration-name* must not contain any special characters in the first eight characters.

   confname

   The first eight characters of *configuration-name*. Zeros at the end of *confname* are not significant characters, i.e. the system does not distinguish between *confname* ABC and *confname* ABC0

   The DBH uses the first eight characters of *configuration-name* to form the names of the following files:

   - session log file (SLF): c*onfname*.SLF
   - temporary realm: c*onfname*.TEMPO.nnn

```
- RLOG files:              confname.RLOG.rlog-time-stamp{1 | 2}[.SAVE]

- session job variable     UDSSQL.DBH.confname
```

The DBH catalogs these files under the user ID with which the user starts the DBH, unless they are user-generated files.

In the case of the independent DBH, the first eight bytes of *configuration-name* are also used to define the communication paths (event names for P1 Eventing and names of the common memory pool); configuration names that are shorter than eight bytes are padded with C'0'. They are used by the database application programs and DCAM administration to establish connections with the DB configuration.

UDS/SQL expands the following files (unless they are large enough anyway):

```
SLF     SPACE = (24,0)

DBSTAT  SPACE = (24,48)

RLOG    SPACE = (192,192)

TEMPO   SPACE = (192,576)
```

Note on the independent DBH

If realms or files of the DB configuration are located on private disks, the disks concerned must be made accessible to several tasks (see the commands manuals for "BS2000 OSD/BC"). This is done before the DBH is started by means of the operator command:

```
/SET-DISK-PARAMETER,...
```

2. By default the DBH creates the temporary user realms on public disk. If the temporary realms are to be held on private disks, the following command must be used to set up a temporary realm in the configuration user ID for each transaction (see PP TRANSACTION in chapter "DBH load parameters"):

```
/CREATE-FILE FILE-NAME=confname.TEMPO.nnn
   ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])
```

confname

The first eight characters of *configuration-name*

nnn

Number of the transaction, three characters with leading zeros

001..<*m*>

m

value of the DBH load parameter PP TRANSACTION (see PP TRANSACTION in chapter "DBH load parameters").

3. By default the DBH creates the session log file (SLF) on public disk. If the session log file is to be held on private disk, it must be set up using the following command:

```
/CREATE-FILE FILE-NAME=confname.SLF
   ,SUPPORT=PRIVATE-*DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])
```

   The SLF is created with a 4-Kbyte page format. If an SLF from a version prior to UDS/SQL V2.0 (i.e. with a 2-Kbyte page format) already exists, and you want to use the same configuration name again, the existing SLF will need to be deleted first.

   > **!** CAUTION!
   > The SLF must not be write-buffered or write/read buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

4. The DB status files must be created if they do not already exist. These files are created with a 4-Kbyte page format and may be placed on different volumes.

   If the DB status files are to reside on private disk, they must be created using the following command:

```
/CREATE-FILE FILE-NAME=confname.DBSTAT[.SAVE]
  ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])
  ,SUPPRESS-ERRORS=*FILE-EXISTING.
```

   > **!** CAUTION!
   > The DB status files must not be write-buffered or write/read buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

5. If the connection module UDSLNKA was linked into the application program, the job switch 28 must be set to OFF so that the linked-in DBH is started (if the job switch 28 is set to ON, the independent variant of the application program is started; see also the “Application Programming” manual, section “Starting a COBOL program”).
6. These commands define the input source from which the DBH is to read in the load parameters. Input sources can be:
   - a SAM or ISAM file
   - a PLAM library element
   - a procedure file (DBH start procedure)

   Below are the respective commands you need, depending on the input source to be assigned:

   a. Assigning a SAM or ISAM file (independent and linked-in DBH)

      `/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=`*`filename`*

   b. Assigning a PLAM library element (independent and linked-in DBH)

      `/ASSIGN-SYSDTA TO=*LIBRARY-ELEMENT(LIB=`*`lib`*`,ELEM=`*`elem`*`,TYPE=S)`
      `/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=SYSDTA`

      See “Note on the linked-in DBH” in chapter "Starting DBH".

   c. Assigning a procedure file (independent DBH)

      If the load parameters are in the DBH start procedure, the following command must be specified before the DBH start command:

```
/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=SYSCMD
```

d. Assigning a procedure file (linked-in DBH)

If the load parameters are in the start procedure for the linked-in application program, the following command must be specified before the application program is started:

```
/ASSIGN-SYSDTA TO=*SYSCMD
```

See “Note on the linked-in DBH” in chapter "Starting DBH".

Note on the linked-in DBH

Input of the load parameters via PPFILE is recommended if SYSDTA is to be used for user data. If the file containing the load parameters is assigned as SYSDTA and the application program also reads from SYSDTA, then:

- when COBOL-DML is used, the timing of load parameter input is dependent upon when the first COBOL module containing subschema declarations was executed:
  - If the data for the program is read in **after** the first branch to a COBOL module containing subschema declarations, the load parameters must come **before** the data for the program.
  - If the data for the program is read in **before** the first branch to a COBOL module containing subschema declarations, the load parameters must come **after** the data for the program.
- When CALL-DML is used exclusively, the timing of load parameter and data input is dependent upon whether the application program reads from SYSDTA before or after the first READYC call.

7. This command allows you to define the variable part of the name of the session job variable. See section “Session job variable”.
8. This command allows you to assign a pubset declaration job variable in which a UDS/SQL pubset declaration is specified. With this declaration you can restrict the requirement that file names should be unique to a selected set of catalog IDs. It is advisable always to assign a pubset declaration job variable in the start procedure as this cannot be done later from another task. If you do not want to restrict the pubsets which are to be taken into account, you can enter the default setting "*" in the pubset declaration job variable. See section “Pubset declaration job variable”.

   ℹ If a faulty job variable assignment exists or an incorrect UDS/SQL pubset declaration is assigned when the DBH starts, this causes the session to abort.

9. These commands are only relevant when the linked-in DBH is started up. In the case of CALL DML application programs, the library from which the SSITAB modules are to be dynamically loaded at execution time must be known. First, the library that was assigned with the link name $UDSSSI is searched. If the SSITAB modules are held in more than one library, e.g. in a separate one for each database, the other libraries with the link names BLSLIB00 through BLSLIB99 can be assigned. Assignment with the link name $UDSSSI and if appropriate also with BLSLIB*nn* is the recommended standard procedure. The abbreviation “$UL” is used for this procedure in the error messages of the product.

   If the link name $UDSSSI is not used or the dynamic-loading operation was unsuccessful, for compatibility reasons a library UDS.MODLIB in the runtime ID or a library assigned with the SET-TASKLIB command is searched. The abbreviation “TSK” is used for this procedure in the error messages of the product.

   With KDBS applications the library containing the PLITAB modules must also be assigned with the link name $UDSPLEX. The dynamic-loading procedure is the same as the handling of the SSITAB modules.

10. If a translation table (UDSTRTAB) is used for user-specific sorting of character fields, the library from which the UDSTRTAB module is to be dynamically loaded must be known. Firstly, a library in the configuration user ID that was assigned with the link name $UDSKONF is searched. For further details see the “Application Programming” manual, section “Translation table for application-specific sorting”.

11. It is recommended that you generally specify which UDS/SQL version is to be used with the SELECT-PRODUCT-VERSION command, as a number of UDS/SQL versions could be installed in parallel and a number of versions of the UDS/SQL subsystem could be preloaded with IMON in the Software Configuration Inventory (SCI). The version number specified should always include the release status and correction status (e.g. 02.9 C00). For further information on using the SELECT-PRODUCT-VERSION command, see section “Using multiple UDS/SQL versions concurrently”. The procedure described in more detail there is also recommended when for long periods only one UDS/SQL version is used on the system.
When the independent DBH is called, the version of the start module UDSSQL (master task) is selected with SELECT-PRODUCT-VERSION; when the linked-in application program is called, the version of the version-dependent connection module LCCONCT to be dynamically loaded is selected. The actual UDS/SQL coding is used in the version appropriate to the start module or linked-in connection module either as a preloaded subsystem or is dynamically loaded from the corresponding SYSLNK library of the UDS/SQL product.
The abbreviation “SCI” is used for this procedure in the error messages of the product. For further details, see section “Using multiple UDS/SQL versions concurrently” and, on linked-in application programs, section “Linking, loading and starting a UDS/SQL-TIAM application program” in the “Application Programming” manual.

    Using UDS-D:

    When UDS-D is used (only possible with an independent DBH), an additional SELECT-PRODUCT-VERSION command should be employed to specify the UDS-D version corresponding to the UDS/SQL version in order to ensure that the correct product parts of UDS-D are loaded dynamically. Here, too, it is recommended that you also use SELECT-PRODUCT-VERSION when for long periods only one UDS-D version is used on the system.

    Private installation

    If you want to use a product version which is not managed in the SCI throughout the system, the following special conditions apply:

    The libraries from which the modules of the product UDS/SQL and, if appropriate, the products UDS-D and UDSKDBS are to be dynamically loaded in the desired version are explicitly assigned with the ADD-FILE-LINK command. The following link names are available for the assignments:

$UDSLIB	Modules of the product UDS/SQL (independent and linked-in DBH)
$UDSDLIB	Modules of the product UDS-D (only with independent DBH)
$UDSKLIB	Modules of the product KDBS (only with linked-in DBH)

    The abbreviation “$UL” is used for this procedure in the error messages of the product.

    For reasons of compatibility, the use of a UDS.MODLIB, possibly with TASKLIB assignment, continues to be supported for dynamically loading the modules if the products are not provided via SCI or via the link names $UDSLIB, $UDSDLIB and $UDSKLIB. The abbreviation “TSK” is used for this procedure in the error messages of the product.

    The SELECT-PRODUCT-VERSION commands for the products UDS/SQL and, if applicable, UDS-D might also be required for a private installation for the following reasons:

    In parallel with a privately installed product version, UDS/SQL and UDS-D subsystems might also be preloaded. The SELECT-PRODUCT-VERSION command ensures that the modules loaded dynamically from the private installation establish a connection to the correct subsystem version.

12. The independent DBH is started with the START-UDS-DBH command.
    We recommend that you specify the RESIDENT-PAGES operand in the START command, as only in this case does UDS/SQL use the more efficient access method FASTPAM (instead of UPAM) for file accesses (for further details, see section “Using FASTPAM in UDS/SQL”).
    As an alternative to the START-UDS-DBH command, for reasons of compatibility the START-EXECUTABLE-PROGRAM command (or the START-PROGRAM command) can also be used to call the independent DBH (for an example, see under “Private installation” in chapter "Starting DBH").
    The master task reads in the DBH load parameters. The DBH initializes the common memory pools and then activates one or more server tasks as ENTER jobs. The master task enters the corresponding ENTER files under the user ID with which the DBH was started (configuration user ID), using the following file names:

    UDS.ENTER.*tsn*.ST0*nn*

    ENTER file(s) for the server task(s)

    tsn
    Task sequence number of the master task

    n
    Number of the server task

    If, when UDS-D is used, PP DISTRIBUTION=START or STANDBY is entered, the following additional ENTER file is created and is activated in response to PP DISTRIBUTION=START:
    UDS.ENTER.*tsn*.CT000

    The master task starts the ENTER jobs for the server tasks (and, in UDS-D operation, for the UDS-D task) as follows:

    - JOB-CLASS = job class of the master task
      if the master task is being run as an ENTER job. If the master task is being run as an interactive task, it starts the ENTER jobs in the default job class for the user ID (JOB-CLASS=*STD).
    - RUN-PRIO = priority of the master task
    - START=IMMEDIATELY
    - START is ignored if it is not permitted in the ADD-USER entry or the job class.
    - TIME=NTL or
      TIME=32000
      if NTL is not permitted in the current user ID or the job class.

    If an attempt to start the ENTER jobs in the job class of the master task is unsuccessful, the master task tries to start them in the default job class for the user ID, but with the same RUN-PRIO and TIME parameters.
    If this, too, is unsuccessful, the tasks are started in the default job class without RUN-PRIO specifications.

    Job classes and user IDs are set up for the database administrator by the BS2000 system administrator.
    You can find out which values are entered for your user ID or job classes by issuing the BS2000 command: SHOW-USER-ATTRIBUTES or SHOW-JOB-CLASS (see the commands manuals for “BS2000 OSD/BC“).

    The master task deletes these ENTER files automatically after normal termination of the DBH.

    Private installation

    If you are using a product version which is not managed in the SCI throughout the system, you cannot as a rule use the START command described above. Use the START-EXECUTABLE-PROGRAM command instead.

    Example

```
/START-EXECUTABLE-PROGRAM FROM-FILE=(
       LIB=$userid.SYSLNK.UDS-SQL.version,ELEM=UDSSQL)
```

13. You load and start the linked-in DBH by calling the application program with the START-PROGRAM command (or with the START-EXECUTABLE-PROGRAM command).

    The preset value NAME-COLLISION=*STD should not be changed, as this may lead to name conflicts in dynamically loadable entries in applications which are linked with the linkage editor BINDER.

    If the application program uses the **COBOL DML**, the connection module calls the modules of the linked-in DBH after the start of the program. The linked-in DBH then reads in the load parameters of the DBH before returning the control to the application program.

    If you are using **CALL DML**, the first READYC call causes the connection module to dynamically load the modules of the linked-in DBH and of the CALL DML translator. The linked-in DBH then reads in the load parameters of the DBH before processing the READYC call.

Messages on starting the DBH

After the ENTER-JOB command to start the server task(s), the master task waits in a time loop for the start messages from the server task(s). Until such time as all tasks have reported to the master task, the following message appears on the database administrator's display terminal and on the operator console at intervals of about 3 minutes:

```
% UDS0212 UDS SYSTEM INITIALIZATION DELAYED. PLEASE CHECK BATCHLIMIT
```

In this case you should check which DBH tasks have not yet been started and why (batch limit, ADD-USER entry), or whether any DBH tasks have terminated abnormally.

The message:

```
% UDS0201 UDS SYSTEM READY
```

indicates that the DBH has been successfully activated.

The session job variable is set to UDSDBH_ACTIVE (see section "Using job variables in UDS/SQL").

UDS/SQL pubset declaration

A UDS/SQL pubset declaration taken into account when the DBH starts (see section "Pubset declaration job variable") is logged as follows:

```
UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS (...,tsn4)
tsn4: UDS-PUBSET-JV: jv-name
tsn4: PUBSETS: catid-group_1
tsn4: PUBSETS: catid-group_2
tsn4: PUBSETS: ...
```

Once each for every catid group specified in the UDS/SQL pubset declaration.

```
tsn4: DEFAULT PUBSET: <default catalog ID of the execution user ID>
tsn4: ------------------------------------------------------------
```

The assignment of a non-existent or an inaccessible job variable or a job variable with the length 0 is rejected as follows:

```
UDS0752 UDS USER ERROR: ACCESS TO UDS-PUBSET-JV VIA JV-LINKNAME
*UDSPS01 FAILED (...,tsn4)
```

Additional information on the cause of the error may be provided in other messages.

Content errors in the UDS/SQL pubset declaration are reported as follows depending on the type of error and the time at which it was detected:

- Syntax errors which were detected during the syntactical analysis (identification of the catid groups):

  ```
  UDS0748 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX): <reason>
  (...,tsn4)
  ```

where <reason> can have the following values:

- CATID GROUP TOO LONG
- TOO MANY CATID GROUPS
- EXCLUDE CATID GROUP NOT ALONE

```
tsn4: UDS-PUBSET-JV: jv-name
tsn4: UDS-PUBSET-JV-CONTENTS:
tsn4: Content of the job variable may be distributed over several lines
tsn4: ------------------------------------------------------------------
```

- Errors detected while the catid groups are being checked using the SDF macro CMDWCC:

```
UDS0749 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX): CATID GROUP
REJECTED BY CMDWCC (...,tsn4)

tsn4: UDS-PUBSET-JV: jv-name
tsn4: PUBSETS: catid-group
```

For each catid group in which the CMDWCC macro detected a fault and resulted in failure, up to 10 times.
If there are more than 10 catid groups with errors, only the first 10 are logged, and the line below is used to show that further catid groups with errors exist:

```
tsn4: FURTHER ERRORS NOT SHOWN
tsn4: ---------------------------------------------------------
```

- Errors detected while the catid groups are being checked using FSTAT:

```
UDS0749 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX): CATID GROUP
REJECTED BY FSTAT (...,tsn4)
tsn4: UDS-PUBSET-JV: jv-name
tsn4: PUBSETS: catid-group, FSTAT-DMS-RC: xxxx
```

For each catid group in which FSTAT detected an error and resulted in failure, up to 10 times.
If there are more than 10 catid groups with errors, only the first 10 are logged, and the line below is used to show that further catid groups with errors exist:

```
tsn4: FURTHER ERRORS NOT SHOWN
tsn4: ---------------------------------------------------------
```

Examples

Example 1: Starting the independent DBH as a batch job

Contents of the DBH start procedure ENTER.DBH.IND:

```
/SET-LOGON-PARAMETERS
/ASS-SYSOUT TO=O.DBH.INT
/SHOW-JV JV.MANUAL.VERSION
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.MANUAL.VERSION
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=&(*VERS)
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=PUBSDECL.PUBS
/SHOW-JV JV-CONTENTS=*LINK(LINK-NAME=UDSPS01)
/CREATE-FILE FILE-NAME=CONFEXMP.DBSTAT
/CREATE-FILE FILE-NAME=CONFEXMP.DBSTAT.SAVE
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONFEXMP
/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=UDSDBB.PP.FILE.IND
/START-UDS-DBH RESIDENT-PAGES=*PAR(MIN=0,MAX=1525)
/EXIT-JOB SYSTEM-OUTPUT=*NONE
```

Start the ENTER procedure:

```
/ENTER-JOB FROM-FILE=ENTER.DBH.IND,JOB-CLASS=JCB32000
JMS0066 JOB 'TRA' ACCEPTED ON 19-01-29 AT 09:28, TSN = 4TMX
```

Log of the ENTER job (excerpt)

```
/SHOW-JV JV.EXAMPLE
% 02.9C00
...
/SHOW-JV JV-CONTENTS=*LINK(LINK-NAME=UDSPS01)
% IUDS
...
/START-UDS-DBH RESIDENT-PAGES=*PAR(MIN=0,MAX=1525)
...
```

Example 2: Starting a linked-in application (COBOL-DML) as an interactive job

Contents of the load parameter file PP.FILE:

```
PP LOG=PUBLIC
PP PARLIST=YES
PP PRIVACY-CHECK=OFF
PP STDCKPT=YES
PP END
```

Contents of the start procedure P.SHIPPING:

```
/BEGIN-PROCEDURE
/SHOW-JV JV.SHIPPING
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.SHIPPING
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=PUBSDECL.PUB
/SHOW-JV JV-CONTENTS=*LINK(LINK-NAME=UDSPS01)
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=&(*VERS)
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING.DBDIR
/ADD-FILE-LINK LINK-NAME=EINART,FILE-NAME=DATA.SHIPMENT.ARTICLE
/ADD-FILE-LINK LINK-NAME=HERST,FILE-NAME=DATA.SHIPMENT.PRODUCER
/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=UDSDBB.PP.FILE
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=LIB.SHIPPING,ELEMENT=SHIPM),  RUN-MODE=ADVANCED
/END-PROCEDURE
```

Call the start procedure:

```
/CALL-PROC FROM=P.SHIPPING
```

Runtime log (excerpt):

```
...
/SHOW-JV JV.SHIPPING
%  02.9C00
...
/SHOW-JV JV-CONTENTS=*LINK(LINK-NAME=UDSPS01)
%  *
...
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=LIB.SHIPPING,ELEMENT=SHIPM)
...
%  UDS0215 UDS STARTING UDS/SQL V2.9 (LINKED-IN), DATE=2019-01-29 (ILL2038,09:27:26/4TE7)
%  UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS (ILL2038,09:27:26/4TE7)
4TE7: UDS-PUBSET-JV:  :IUDS:$XXXXXXXX.PUBSDECL.ALL
4TE7: PUBSETS:        *
4TE7: DEFAULT PUBSET: IUDS
4TE7: ----------------------------------------
%  UDS0722 UDS ORDER ADD RLOG 190129082726 IN EXECUTION (ILL1283,09:27:27/4TE7)
%  UDS0354 UDS ALOG CHECKPOINT FOR SHIPPING (ILL1307,09:27:27/4TE7)
4TE7: ALOG-CKPT 20190129082725: CKPT ALREADY EXISTING.
4TE7: MAXDB          =      1
4TE7: TRANSACTION    =      1
4TE7: SUBSCHEMA      =      1
4TE7: 2KB-BUFFER-SIZE=      1
4TE7: 4KB-BUFFER-SIZE=      1
4TE7: 8KB-BUFFER-SIZE=      0
```

```
4TE7: LOG            = PUBLIC
4TE7: LOG-2          = NO
4TE7: LOG-SIZE       = (    192,    192)
4TE7: RESERVE        = NONE
4TE7: WARMSTART      = STD
4TE7: CONSOLE        = NO
4TE7: STDCKPT        = YES
4TE7: LOCK           = STD
4TE7: PRIVACY-CHECK  = OFF
4TE7: CONFNAME       = $XXXXXXXX.SHIPPING
4TE7: DATABASES OF CONFIGURATION:
4TE7:   $XXXXXXXX.SHIPPING                      ,EXCLUSIVE-UPD     ,*SYSTEM
%  UDS0356 UDS EXECUTION OF ORDERS FOR SHIPPING TERMINATED (ILL1309,09:27:27/4TE7)
DB-ART-NO 000001
DB-NAME SUMMER DRESS WITH JACKET
DB-ART-NO 000002
DB-NAME TWO-PART SUMMER DRESS
DB-ART-NO 000003
DB-NAME WHEAT BEER
DB-ART-NO 000004
DB-NAME LAGER
DB-ART-NO 000005
DB-NAME LEMONADE
DB-ART-NO 830950
DB-NAME PLEATED DRESS WITH JACKET
DB-ART-NO 830993
DB-NAME JERSEY CREPE DRESS
DB-ART-NO 835758
DB-NAME FLOWING JERSEY DRESS
DB-ART-NO 831213
DB-NAME FLOWING JERSEY DRESS WITH JACKET
DB-ART-NO 835928
DB-NAME T-SHIRT DRESS
DB-ART-NO 835952
DB-NAME POLO DRESS
DB-ART-NO 001140
DB-NAME LOW-FAT NATURAL YOGHURT
DB-ART-NO 001141
DB-NAME ORGANIC NATURAL YOGHURT
*** NORMAL PROGRAM END ***
%  UDS0354 UDS ALOG CHECKPOINT FOR SHIPPING (ILL1307,09:27:27/4TE7)
4TE7: ALOG-CKPT 20190129082727: FIXED ( ALOG-NR:000000001, START-CKPT: 20190129082725 ).
%  UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE  (ILL1758,09:27:27/4TE7)
 4TE7: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 4TE7: ------------------------------------------------------------------------
 4TE7: SHIPPING                 115       1491        103        696        106
%  UDS0213 UDS NORMAL SYSTEM TERMINATION WITH **************115 DML-STATEMENTS 2019-01-29
(ILLY033,09:27:27/4TE7)
```

2.3.2 Defining and altering the DB configuration

The user defines a DB configuration by telling the DBH which databases are to be attached. This can be done in the following ways, depending on the type of DBH used:

linked-in DBH

- using DBH load parameters

independent DBH

- using DBH load parameters and
- using DAL commands

In mono-DB operation, the configuration name may be identical to the name of the database (cf. load parameter DBNAME, "DBH load parameters").

The **independent DBH** can also be started with an "empty DB configuration" if no databases are specified in the load parameters on starting the DBH. It is possible to use DAL commands to attach databases at a later stage.
A DB configuration may be altered during a session or after a session abort.

DAL commands can be used to **attach databases (ADD)** and **detach databases (DROP)**; see section “The Database Administrator Language DAL”.

Altering the DB configuration during the session

To attach or detach databases to or from a DB configuration, a request to attach or detach one or more databases is sent to the DBH in the form of the DAL command ADD or DROP. This request is not implemented until the DAL command PERFORM is issued.

If the database which is to be attached is inconsistent, the DBH executes a warm start before making it accessible to transactions.

This warm start comprises:

- rollback of all uncompleted transactions in the database
- rerunning transactions which have made changes not yet implemented in the database

The SLF maintains a fail-safe log of the current DB configuration as a basis for an automatic session restart after a session abort. Consequently, the DBH obtains the information required to perform a session restart from the SLF. This information includes the currently applicable DBH load parameters and the currently attached databases.

Note on UDS-D

If the DB configuration is altered during the session, the database administrator must adapt the distribution table to match via the administration interface (DAL commands). If there are still transactions in the PTC (prepared to commit) condition even after a database warm start, the associated database cannot be detached from the DB configuration (see section “Altering the DB configuration: notes for UDS-D”).

Altering the DB configuration after a session abort (reconfiguration)

After a session abort, the database administrator may dispense with a session restart and start a new session instead with a new DB configuration setup, new DBH load parameters, the same (or a different) configuration name, and any DBH variant.

Reconfiguration is typically useful when a session abort occurs shortly before the scheduled DBH end, i.e. when there is no point in resuming the session by means of a session restart.

To carry out a reconfiguration, the database administrator deletes the SLF file of the aborted session (see section “Session log file (SLF)”), thereby declaring the aborted session terminated.

Reconfigurations are subject to a number of restrictions (see “Notes on reconfiguration” in chapter "DBH load parameters").

> ¡ After a session abort, the database administrator may also use an earlier status of the databases as a fallback. However, the database administrator then assumes responsibility for consistency relations between the databases.

2.3.3 Terminating the DBH

The independent DBH can be terminated by means of the DAL commands:

- CLOSE RUN-UNITS:
  Normal termination of the DBH
- CLOSE CALLS:
  Fast normal termination of the DBH
- %TERM:
  Immediate abortion of the DBH

Normal termination of the DBH

Example

UDSADM statement:

```
01 //EXECUTE-DAL-CMD CMD=CLOSE RUN-UNITS
```

Messages:

```
    %   UDS0220 UDS RECEIVED COMMAND: CLOSE RUN-UNITS (OPOX073,09:35:23/4TUC)
02  %   UDS0241 UDS TERMINATION INITIATED ON TRANSACTION (OPCF024,09:35:23/4TUC)
    %   UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:35:23/4TUC)
03  %   UDS0241 UDS TERMINATION INITIATED ON SERVERTASK (OPO0024,09:35:29/4TUC)
    %   UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE  (OPCC758,09:35:30
/4TUC)
     4TUC: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
     4TUC: ------------------------------------------------------------------------
     4TUC: CUSTOMER                   6         68         37         23         23
04  %   UDS0213 UDS NORMAL SYSTEM TERMINATION WITH ***************7 DML-STATEMENTS 2019-01-
29 09:35:30 (OPOB033,09:35:30/4TUC)
```

01 The DAL command:

   ```
   CLOSE RUN-UNITS
   ```

   starts the normal termination procedure.

02 No more transactions are permitted.

03 Once all transactions have been terminated, the master task terminates all server tasks.

04 Once all server tasks have been terminated, the master task invalidates the session log file (SLF), deletes the temporary realms and deletes the RLOG files that belong to this configuration name and that are logically empty.

Note on UDS-D/openUTM

CLOSE RUN-UNITS is rejected if there are still secondary subtransactions or openUTM transactions in the PTC state.

The database administrator can nonetheless terminate the DBH normally, provided the PTC state is terminated first (see section “Terminating the PTC state”).

Fast normal termination of the DBH

Example:

UDSADM statement:

```
01 //EXECUTE-DAL-CMD CMD=CLOSE CALLS
```

Messages:

```
    %  UDS0220 UDS RECEIVED COMMAND: CLOSE CALLS (OPOX073,09:29:54/4TMW)
    %  UDS0241 UDS TERMINATION INITIATED ON TRANSACTION (OPCF024,09:29:54/4TMW)
02  %  UDS0348 UDS CANCEL ALL TRANSACTIONS STARTED  (OPC1035,09:29:54/4TMW)
    %  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:54/4TMW)
03  %  UDS0241 UDS TERMINATION INITIATED ON SERVERTASK (OPO0024,09:30:00/4TMW)
    %   UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE  (OPCC758,09:30:01
/4TMW)
     4TMW: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
     4TMW: ------------------------------------------------------------------------
     4TMW: SHIPPING                   1         88         58         24         24
     4TMW: CUSTOMER                   0         35         22          8          8
04  %  UDS0213 UDS NORMAL SYSTEM TERMINATION WITH ***************1 DML-STATEMENTS 2019-01-29
09:30:01 (OPOB033,09:30:01/4TMW)
```

01 The DAL command:

   `CLOSE CALLS`

   accelerates termination of the DBH.

02 FINISH WITH CANCEL is enforced upon all uncompleted transactions, i.e. all changes made during these transactions are rolled back with the aid of the associated before-image files.

   The DBH passes status code 122 or 151 to application programs whose transactions were aborted with FINISH WITH CANCEL.

03 The master task then attempts to terminate the DBH normally.

04 The session is not held to be terminated until the master task has deleted the contents of the SLF.

Note on UDS-D/openUTM

CLOSE CALLS is rejected if there are still secondary subtransactions or openUTM transactions in the PTC condition.

The database administrator can nonetheless terminate the DBH normally, provided the PTC state is terminated first (see section “Terminating the PTC state”).

Immediate abortion of the DBH

Example

UDSADM statement:

```
01 //EXECUTE-DAL-CMD %TERM
```

Messages:

```
    %  IDA0N51 PROGRAM INTERRUPT AT LOCATION '0003AD70 (SCCDUMP), (CDUMP), CODE=UDS'
02  %  IDA0N53 DUMP BEING PROCESSED. PLEASE HOLD ON
    %  IDA0N54 'USERDUMP' WRITTEN TO FILE '$XXXXXXX.DUMP.DBH.4TMW.00006'
    %  IDA0N55 TITLE: 'TSN-4TMW  UID-XXXXXXX  AC#-Z1285  USERDUMP  PC-00034D98  EC-50  VERS-
120 DUMP-TIME 2019-01-29 09:30:01'
    %  UDS0220 UDS RECEIVED COMMAND: %TERM (OPOX073,09:30:01/4TMW)
03  %  UDS0202 UDS ABNORMAL SYSTEM TERMINATION WITH **************23 DML-STATEMENTS 2019-01-
29 09:30:01 (OPY8003,09:30:01/4TMW)
    %  EXC0732 ABNORMAL PROGRAM TERMINATION. ERROR CODE 'NRT0101': /HELP-MSG NRT0101
```

01 The DAL command:

   ```
   %TERM
   ```

   dispenses with the normal termination activities of the DBH and terminates the session by the shortest route (emergency halt).

02 The DBH places a memory dump of its main memory area in the file DUMP.*jobname*.*tsn*.*nnnnn* (when PP DUMP=ALL) or SYS.ADUMP.*jobname*.*tsn*.*nnnnn* (when PP DUMP=STD).

03 The master task aborts the session; the databases of the DB configuration are marked as inconsistent if updates have been performed since the last consistency point. The interrupted session can be resumed by means of a session restart.

   The session is not held to be terminated until the SLF or its contents have been deleted.

2.4 Parallel database operation

UDS/SQL supports parallel database operation, i.e. several Database Handlers, independent and/or linked-in, can be active simultaneously in one host. (For information on the parallel use of different DBH versions on one host, see section “Using multiple UDS/SQL versions concurrently”).

It is also possible to load the DBH several times under the same user ID for different databases or DB configurations, and for several DBHs running in parallel to access a single database for reading, i.e. by means of the SHARED-RETRIEVAL option in the load parameter PP DBNAME or in the DAL command ADD DB. This also applies to shadow databases.

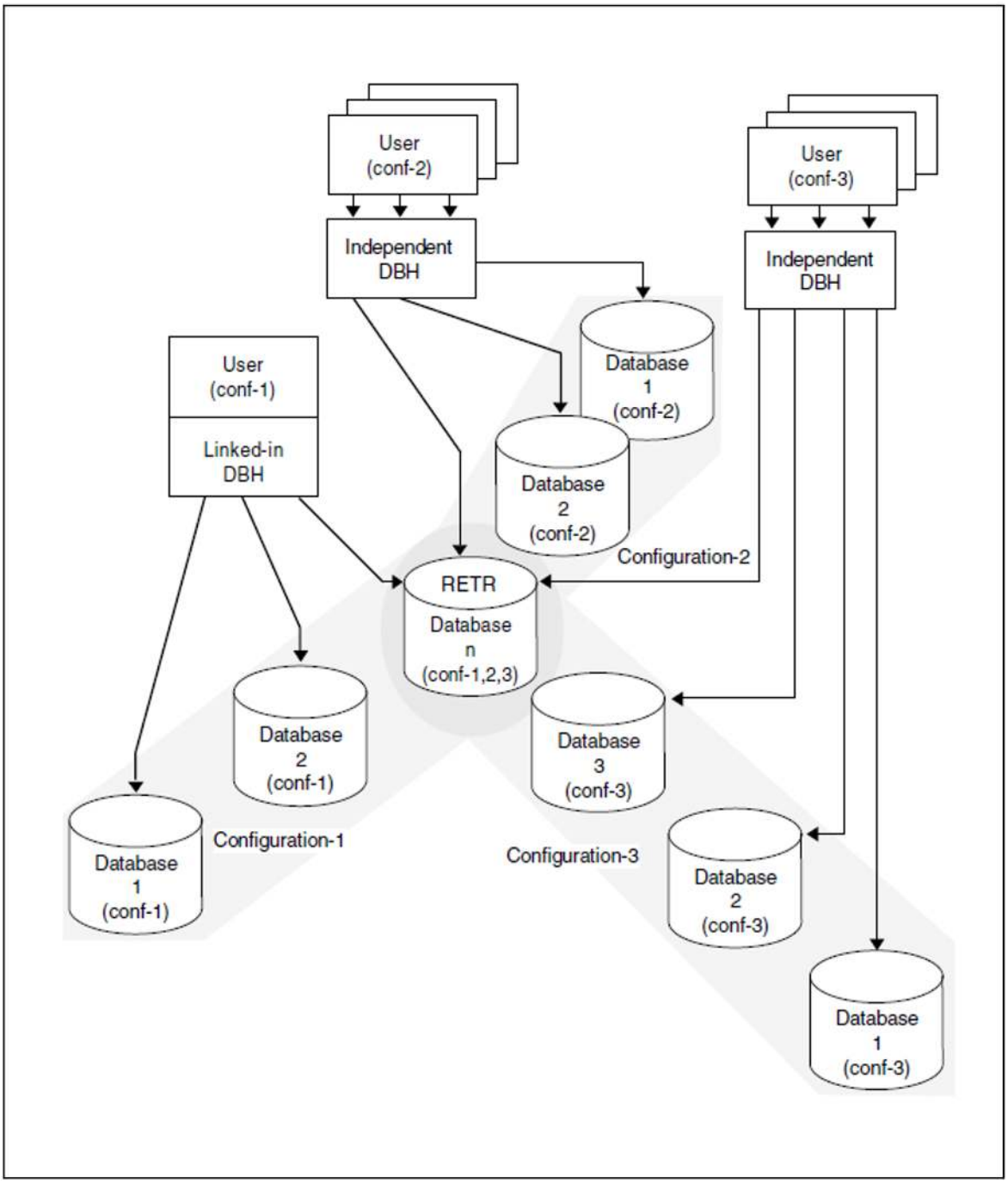


Figure 8: Example of parallel database operation

3 DBH load parameters

The DBH load parameters define the basic characteristics of a session. With the exception of PP DBNAME and PP PASSWORD, no load parameter may be specified more than once.There are default values for any DBH load parameters left unspecified. Only PP END and PP LOG must be specified. You use PP END to terminate the input and PP LOG to determine whether or not the DBH is to maintain an RLOG file.

The DBH load parameter names may not be truncated, with the exception of a number of operands. The mandatory character string is underscored.

The operands must not be separated by blanks, since every entry after the first blank is interpreted as a comment.

Example

```
PP DBNAME=dbname,'BLANK' n
```

In this case, the value ”*n*“ will be interpreted as a comment.

Input records which begin with "#" or "*" are interpreted as comments.

All DBH load parameters may be used both for the linked-in DBH and for the independent DBH. If load parameters which are not evaluated in the current DBH variant are specified, a warning is issued.

Evaluation of DBH load parameters

The specified DBH load parameters are evaluated only on starting the DBH. The values which apply in the event of a session restart are those which were present in the SLF at the time of a session abort: some of the session characteristics may have been redefined by the database administrator in the course of the session using the Database Administrator Language (DAL).
The DBH indicates by means of the following message that it skips the specified DBH load parameters in the event of a session restart:

```
% UDS0709 UDS SESSION RESTART, PROGRAM-PARAMETERS WILL BE SKIPPED.
```

It is possible, however, to modify the default procedure and respecify the DBH load parameters explicitly after a session abort. This entails deleting the contents of the SLF (see section “Session log file (SLF)”).Once the SLF file has been deleted, the DBH starts a new session and evaluates the DBH load parameters.

Responses in error situations

- If a DBH load parameter is entered more than once (except in the case of PP DBNAME or PP PASSWORD), the DBH evaluates only the first correct parameter and ignores the rest.
- If the DBH fails to identify the keyword, it issues an error message and aborts the DBH start.
- Load parameters must be no more than 72 characters in length. Longer character strings are rejected by the DBH and treated as syntax errors.
- If DBH load parameters are entered interactively, the DBH issues an error message in the event of syntax errors. In such cases, the error can be corrected by re-entering the load parameter correctly. If this is not done, the DBH aborts the DBH start after PP END.

Exception

No direct corrections are possible in the case of PP PASSWORD and PP DBNAME. If the error that occurred for PP DBNAME is a syntax error, however, you may specify additional statements for PP DBNAME, but these will only be checked for syntax. The DBH start is aborted by the DBH after PP END even if valid statements for PP DBNAME are present.

- If stored DBH load parameters are entered from a cataloged file, direct error correction is not possible. In the event of an error, the DBH aborts the DBH start. The error must be corrected in the file.
- If DBH load parameters containing numeric values outside the permitted ranges are entered, the DBH assumes the highest or lowest permitted value, as appropriate, and issues a warning for the corresponding load parameter.

The **independent DBH** processes the following load parameters.

Parameter	Default value	Meaning
PP 2KB-BUFFER-SIZE=*n*	1	Defines size of the 2-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 4KB-BUFFER-SIZE=*n*	1	Defines size of the 4-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 8KB-BUFFER-SIZE=*n*	0	Defines size of the 8-Kbyte system buffer pool in Mbytes; *n*=0 or *n*=3..2047
PP ADM={REMOTE | LOCAL}	REMOTE	REMOTE: Administration from any terminal via DCAM LOCAL: The master task reserves the terminal permanently. Administration is possible only via the database administrator's terminal or the console.
PP ADMPASS=*admpassword*	-	Defines password for administration via DCAM ENTRY: not permitted
PP BCAM-PREFIX=*prefix*	SUD$	Defines a prefix for names of user tasks which execute on virtual hosts
PP CATPASS={STANDARD | password}	STD	Defines password for files created by the DBH, such as ALOG files and temporary realms
PP CHCKTIME=*n*	60	Specifies period in seconds for connection and transaction checking by the UDS-D task *n*=60..900
PP CP-SIZE=*n*	1024	Specifies minimum size in Kbytes of common pool; *n*=1..16384
PP CPU={MONO-PROCESSOR | MULTI-PROCESSOR}	MONO-PROCESSOR	Specifies processor type used
PP CUP-SIZE=*n*	1024[1] 128 [2]	Specifies minimum size in Kbytes of communication pool; *n*=1..16384

PP DBNAME=[$*userid.*]*dbname* [.*copyname*] [,SHARED-RETRIEVAL] [,{n | [*n*],*bufferid*}]	-	Names databases in DB configuration; max. 222 databases *n*: defines size of exclusive buffer pool in Mbytes; *n*=0..2047 *bufferid:* buffer pool identifier
PP DEACT={YES | NO}	YES	Enables deactivation of UDS/SQL tasks due to computer workload
PP DEADTIME=*n*	60	Time in seconds to resolve interconfiguration deadlocks or deadlocks involving openUTM *n*=5..900
PP DIP-SIZE=n	1024[1] 64[2]	Specifies minimum size in Kbytes of distribution pool; n=1..16384
PP DISDB=*n*	1	Maximum number of remote databases that can be accessed per transaction; *n*=1..32
PP DISTABLE=[:*catid*:][$*userid.*] *file-name*	-	Specifies input file for creating distribution table
PP DISTRIBUTION={NO | STANDBY | START}	NO	Controls participation in UDS-D operation NO UDS-D operation not possible STANDBY UDS-D operation is being prepared and can be started later with &START DISTRIBUTION START UDS-D operation is started
PP DUMP={STD | ALL}	ALL	Determines the scope of a dump
PP END	-	Terminates input of load parameter
PP IO={ASYNC | SYNC}	ASYNC	Executes I/Os in server tasks asynchronously or synchronously
PP LOCK={STD | SHARED | EXCLUSIVE}	STD	Defines locking protocol

PP LOG={NO | :*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	-	Maintains the RLOG file
PP LOG-2={:*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	-	Maintains the duplicate RLOG file
PP LOG-SIZE=([*primary*] [,[*secondary*]])	192,192	Specifies amount of storage (number of PAM pages) in RLOG files
PP MAXDB=*n*	Sum of PP DBNAME	Specifies maximum number of databases in DB configuration; *n*=1..122
PP MPSEG={STD | 64K}	STD	Specifies segment size
PP ORDER-DBSTATUS={STD | SPECIAL}	STD	Controls system behavior if new UPDATE transactions or processing chains collide with the execution of pending requests
PP PARLIST={YES | NO}	NO	Lists parameters used
PP PASSWORD={NONE | STANDARD | *kennwort* (*kennwort*,*kenn-wort*,...)}	STD	Defines password that the DBH must use along with PP CATPASS in order to open files
PP PRIVACY-CHECK={STD | NO-KSET | OFF}	STD	Controls handling of privacy checks
PP PTCSYNCH= ([{WAIT | ABORT | COMMIT}] [,[{WAIT | ABORT | COMMIT}]])	WAIT,WAIT	Controls handling of transactions in PTC state; the first value applies to warm starts, the second to the current session if the state of the primary subtransaction cannot be ascertained

PP RESERVE= {NONE | :*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	NONE	Specifies reserve volumes for RLOG files
PP RESULT-DELAY=*n*	0	Groups together request results for user tasks; *n*=1..*m* *m*=PP TRANSACTION
PP SCHEDULING={SYMMETRIC | ASYMMETRIC}	SYMMETRIC	Controls optimization of communications between the user task and server task and the processing of pending DML jobs in the server task
PP SERVERTASK=*n*	1	Defines number of server tasks under independent DBH *n*=1..30
PP SIP-SIZE=*n*	1024[1] 128[2]	Specifies size of the SSITAB pool in Kbytes; *n*=1..16384
PP SQL=*n*	4	Defines maximum number of simultaneously active SQL conversations; *n*=0...9999
PP SQL-LIMIT=*n*	10	Sets minimum time for which UDS/SQL maintains the conversationspecific data for inactive conversations; *n*=5..999 minutes
PP STDCKPT={YES | NO}	NO	Writes standard checkpoints in AFIM logging at a DBH start, DBH end, and a session restart
PP SUBSCHEMA=*n*	1	Defines maximum number of subschemas that can be used at one time in a database; *n*=1..100
PP SUBTRANSACTION=*n*	0	Defines maximum number of logical file names open at one time per database (KDBS only); *n*=1..254
PP TA-ACCESS={STD | SHARED}	STD	Defines usage modes for transactions

PP TRANSACTION={*n* | ([*n*][,*m*])}	(4,1)	*n*: maximum number of simultaneously active transactions and user tasks *n* = 1..225; *m*: maximum number of secondary subtransactions that this DBH can process simultaneously *m* = 1..*n* and *m* <= *n*
PP UCON=C'{(*mn*) | <*x* | *nnnn*}' [,{MSG | UDS}]	C' <U'	Defines operator terminal (UCON) on which DCAM administration is to be logged
PP WAIT={EVENT | BUSY}	EVENT	Sets wait mode
PP WARMSTART={STD | FAST | VERY-FAST}	STD	Determines duration of a warm start

Table 8: Load parameters of the independent DBH

[1] if PP MPSEG=STD is specified

[2] if PP MPSEG=64K is specified

The **linked-in DBH** processes the following load parameters:

Parameter	Default value	Meaning
PP 2KB-BUFFER-SIZE=*n*	1	Defines size of the 2-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 4KB-BUFFER-SIZE=*n*	1	Defines size of the 4-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 8KB-BUFFER-SIZE=*n*	0	Defines size of the 8-Kbyte system buffer pool in Mbytes; *n*=0 or *n*=3..2047
PP CATPASS={STANDARD | *password*}	STD	Defines password for files to be created by DBH, such as ALOG files and temporary realms
PP CONSOLE={YES | NO}	NO	Outputs linked-in DBH messages to operator console as well
PP DBNAME=[$*userid.*]*dbname* [.*copyname*] [,SHARED-RETRIEVAL] [,{n | [*n*],*bufferid*}]	-	Names databases in the DB configuration; max. 222 databases *n*: defines size of user buffer pool in Mbytes; *n*=0..2047 *bufferid*: buffer pool identifier
PP END	-	Terminates input of load parameters
PP LOG= {NO | :*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	-	Maintains the RLOG file
PP LOG-2= {:*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	-	Maintains the duplicate RLOG file

PP LOG-SIZE=([*primary*] [,[*secondary*]])	192,192	Specifies amount of storage (number of PAM pages) in RLOG files
PP MAXDB=*n*	Sum of PP DBNAME	Defines maximum number of databases in DB configuration; *n*=1..222
PP PARLIST={YES | NO}	NO	Lists load parameters used
PP PASSWORD={NONE | STANDARD | *password* (*password*, *password*,...)}	STD	Defines password that the database must use along with PP CATPASS in order to open files
PP PRIVACY-CHECK={STD | NO-KSET | OFF}	STD	Controls handling of privacy checks
PP RESERVE= {NONE | :*catid*: | PUBLIC (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/device-3*]]) (*vsn-1*[,*vsn-2*[,*vsn-3*]])}	NONE	Specifies reserve volumes for the RLOG files
PP STDCKPT={YES | NO}	NO	Writes standard checkpoints in AFIM logging at a DBH start, DBH end, and a session restart
PP SUBSCHEMA=*n*	1	Defines maximum number of subschemas that can be used at one time in a database; *n*=1..100
PP SUBTRANSACTION=*n*	0	Defines maximum number of logical file names open at one time per database (KDBS only); *n*=1..254
PP TRANSACTION=1	1	Defines number of simultaneously active transactions
PP WARMSTART={STD | FAST | VERY-FAST}	STD	Determines duration of a warm start

Table 9: Load parameters of the linked-in DBH

The value of the load parameter PP TRANSACTION is fixed; any attempt to define a different value is ignored. It is therefore not necessary to specify this parameter.

Defining the 2-Kbyte system buffer pool size (2KB-BUFFER-SIZE)

```
PP 2KB-BUFFER-SIZE=n
```

Default value:

1

n Size, in Mbytes, of the system buffer pool in which realm pages of databases with a 2-Kbyte page format are buffered:

n = 1..2047

The size of the system buffer pool determines how often the DBH must access realms of the databases belonging to the DB configuration. Selecting a large buffer size will reduce the number of accesses required. Depending on the available main memory, however, this may increase the paging rate and thus delay database operations in certain circumstances.The start phase of UDS/SQL is also prolonged, as is the setting of consistency points and database reconstruction (see also the load parameter “PP WARMSTART” in chapter "DBH load parameters").

For the specified value to be meaningful, the system configuration must have the necessary resources. It is not a good idea to use the maximum value, since all the available address space would then be taken up by the system buffer pool alone.

Defining the 4-Kbyte system buffer pool size (4KB-BUFFER-SIZE)

```
PP 4KB-BUFFER-SIZE=n
```

Default value:

1

n Size, in Mbytes, of the system buffer pool in which realm pages of databases with a 4-Kbyte page format are buffered:

n = 1..2047

The size of the system buffer pool determines how often the DBH must access realms of the databases belonging to the DB configuration. Selecting a large buffer size will reduce the number of accesses required. Depending on the available main memory, however, this may increase the paging rate and thus delay database operations in certain circumstances.The start phase of UDS/SQL is also prolonged, as is the setting of consistency points and database reconstruction (see also the load parameter “PP WARMSTART” in chapter "DBH load parameters").

For the specified value to be meaningful, the system configuration must have the necessary resources. It is not a good idea to use the maximum value, since all the available address space would then be taken up by the system buffer pool alone.

Defining the 8-Kbyte system buffer pool size (8KB-BUFFER-SIZE)

```
PP 8KB-BUFFER-SIZE=n
```

Default value:

0

n Size, in Mbytes, of the system buffer pool in which realm pages of databases with an 8-Kbyte page format are buffered:

n = 0 or

n = 3..2047

If no databases with an 8-Kbyte page format are to be processed in the session, you may optionally prevent the creation of an 8-Kbyte system buffer pool by omitting the load parameter entirely or by specifying PP 8KB-BUFFER-SIZE=0.

The size of the system buffer pool determines how often the DBH must access realms of the databases belonging to the DB configuration. Selecting a large buffer size will reduce the number of accesses required. Depending on the available main memory, however, this may increase the paging rate and thus delay database operations in certain circumstances.The start phase of UDS/SQL is also prolonged, as is the setting of consistency points and database reconstruction (see also the load parameter “PP WARMSTART” in chapter "DBH load parameters").

For the specified value to be meaningful, the system configuration must have the necessary resources. It is not a good idea to use the maximum value, since all the available address space would then be taken up by the system buffer pool alone.

Defining DBH administration (ADM)

Independent DBH only!

```
PP ADM={REMOTE | LOCAL}
```

Default value:

REMOTE

REMOTE

DBH administration via DCAM is possible either using the UDSADM administration program (recommended) or by connecting a display terminal directly to DCAM (see section “Administration of UDS/SQL via UDSADM” and section “Administration of UDS/SQL via DCAM”). A DCAM application is opened at UDS/SQL initialization. The application name is identical with the configuration name or the first 8 characters of *configuration-name*.

LOCAL

The master task reserves the terminal permanently or administration is effected from the console using the INFORM-PROGRAM command. No DCAM application is opened.

Defining a password for administration via DCAM (ADMPASS)

Independent DBH only!

```
PP ADMPASS=admpassword
```

admpassword

Defines a password for administration if PP ADM=REMOTE has been specified (see the section “Administration of UDS/SQL via UDSADM” and section “Administration of UDS/SQL via DCAM”). It may be up to four bytes in length and is represented as follows:

C’*xxxx*’:
one through four alphanumeric characters

X’*nnnnnnnn*’:
one through eight hexadecimal digits

The value of PP ADMPASS can be changed dynamically with the DAL commands ADD (see "Attaching databases, realms and passwords (ADD)") and DROP (see "Detaching databases, realms and passwords (DROP)").

> If you wish to use lowercase letters, you should select the X format, since some input interfaces (e.g. INFORM-PROGRAM) convert lowercase letters into uppercase letters.

Defining a prefix for user tasks in UDS-D distributions (BCAM-PREFIX)

Independent DBH only!

```
PP BCAM-PREFIX=prefix
```

Default value:

SUD$

prefix Defines a prefix for BCAM names for distributed communication between the user task and the remote UDS/SQL configuration. This prefix is required specifically when different virtual hosts are defined in a BS2000 system and multiple UDS/SQL configurations are to be distributed over virtual hosts (see also section “Using the UDS/SQL configuration on a virtual host”).

prefix must be four characters long and ensure that the BCAM names generated with it are permitted names and that these differ from the names of other BCAM applications in their first four characters.

The prefix UDSD is not permitted because it is reserved for the downwardcompatible connection to DCAM_BAS configurations.

If a value which differs from the default value is used for the load parameter BCAM-PREFIX, you are recommended to employ at least UDS-D V2.6 on all remote configurations which participate via UDS-D - also in participating configurations which do not execute on virtual hosts. Otherwise internal bottlenecks which can also hinder other DML processing can occur on configurations with older UDS-D versions.

Defining a password for files to be created by the DBH (CATPASS)

```
PP CATPASS={STANDARD | password}
```

Default value:

STANDARD

STANDARD

UDS/SQL protects newly created files with a default read password: C'UDS'BLANK''

password

The password may be up to four bytes in length and is represented as follows:

C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

d:
where *d* is a decimal number (comprising up to eight digits and a sign) to be converted to a binary value

The syntax conventions for BS2000 apply (see the description of the ADD-PASSWORD command in the commands manuals of "BS2000 OSD/BC").

The PP CATPASS parameter is used to specify a read password to protect all those files which are created by the DBH as required, such as ALOG files and temporary realms. The specified password is applicable to all these files. The password for ALOG files created by the utility routines is C'UDS'BLANK''.

The DB status files created by the user and the session log file (SLF) are **always** protected by C'UDS'BLANK'', regardless of any PP CATPASS specification. This password is not designed to protect the contents of the files against unauthorized access; it is used to prevent these files which are employed in various sessions from being inadvertently deleted. These files only contain database management data. In the case of DB status files it is possible that status files external to the configuration will be needed for openUTM status requests to UDS/SQL. In this case the default password precludes the possibility of conflicting passwords (see section "Assigning passwords to UDS/SQL files"). In the case of the SLF the default password is required to enable the DBH to access the SLF even before it has evaluated the DBH load parameters.

The password C'UDS'BLANK'' and read passwords specified via PP CATPASS, PP PASSWORD or the DAL command ADD PW should not be used for other files. This is because UDS/SQL issues these passwords internally, so in special situations files would become accessible which users want to protect.

The RLOG files are protected by a password which is derived from the timestamp of the RLOG file.

The password is designed above all to prevent inadvertent deletion. As the RLOG files are extremely important for ensuring the consistency of the data during ongoing operation, each RLOG file is protected by a password of its own. Generally the DBH deletes RLOG files when they are no longer required. If in exceptional circumstances an RLOG file has to be deleted for other reasons, you can issue the password using the RLOGPASS utility routine in the UDS-SQL-T delivery unit.

Monitoring connections and transactions (CHCKTIME)

For UDS-D

```
PP CHCKTIME=n
```

Default value:

60

n Period in seconds in which the UDS-D task checks whether any of the server tasks is still processing a DML statement. If this is the case the UDS-D task also sends a timing acknowledgment to user tasks waiting for a response to a DML statement still being processed in a remote CPU. This acknowledgment indicates that the DML statement is still being processed and that the logical connection is still being maintained.

The period set with PP CHCKTIME is also used for a number of other monitoring functions (see section “Time-controlled monitoring of connections and transactions”).

n = 60..900

PP CHCKTIME is operative in UDS-D mode only.

Displaying linked-in DBH messages on the operator console (CONSOLE)

Linked-in DBH only!

```
PP CONSOLE={YES | NO}
```

Default value:

NO

YES UDS/SQL is to output all messages to the operator console as well. This applies irrespective of whether the linked-in DBH was started as an interactive task or as an ENTER job.

NO Once the DBH load parameters have been read in, UDS/SQL is to output all messages only to the display terminal from which the linked-in DBH was started (or in the corresponding SYSOUT log in the case of batch jobs).

Specifying the minimum size for individual common pools (CP-SIZE)

Independent DBH only!

```
PP CP-SIZE=n
```

Minimum value:

1024 Kbytes

n Size of a common pool in Kbytes

n = 1..16384

n is rounded up to the nearest integral multiple of 1024.

If *n* > 16384, the DBH uses 16384 Kbytes.

Common pools are created dynamically in a session. The individual common pools are created with size given in PP CP-SIZE or, if this minimum value is insufficient, by using the size value required by the DBH.

The minimum size of a common pool can be displayed by means of the load parameter PP PARLIST or as the CP-SIZE value of the DAL command DISPLAY PP.

Specifying the processor type (CPU)

Independent DBH only!

```
PP CPU={MONO-PROCESSOR | MULTI-PROCESSOR}
```

Default value:

MONO-PROCESSOR

MONO-PROCESSOR

The UDS/SQL session is to run on a monoprocessor CPU.

MULTI-PROCESSOR

The UDS/SQL session is to run on a multiprocessor CPU.

The PP CPU parameter is used to specify whether UDS/SQL is to run on a host computer with a monoprocessor or a multiprocessor CPU. On the basis of this information UDS/SQL selects the best holding mechanism to enable it to serialize its internal processing sequences. If PP CPU is incorrectly specified, runtime performance is likely to be impaired.

Specifying the minimum size for the communication pool (CUP-SIZE)

Independent DBH only!

```
PP CUP-SIZE=n
```

Minimum value:

1024 Kbytes if PP MPSEG=STD is specified
128 Kbytes if PP MPSEG=64K is specified.

n Size of communication pool in Kbytes

n = 1..16384

n is rounded up to the nearest integral multiple of 1024 for PP MPSEG=STD and up to the nearest integral multiple of 64 for PP MPSEG=64K. For PP MPSEG=64K, values above 8192 are not advisable.

If *n* > 16384, the DBH uses 16384 Kbytes.

If the storage space requirement calculated by the DBH exceeds the minimum or the specified value, the value computed by the DBH is used when creating the communication pool.

If PP MPSEG=STD, the communication pool is created in the upper address space of the UDS/SQL tasks (> 16 MB). In user tasks the communication pool is created in the lower or upper address space depending on AMODE (24/31). If PP MPSEG=64K, the communication pool is located in the lower address space in all tasks. If applications in AMODE 24 are also used in the session, the size of the communication pool must be set such that it fits into the lower address space of this application.

The size of the actually created communication pool can be obtained with the load parameter PP PARLIST or from the CUP-SIZE value output by the DAL command DISPLAY PP. If the average subschema size or job length of an SQL statement exceeds 16 Kbytes, you should increase the actual size by using PP CUP-SIZE.

Naming databases (DBNAME)

```
PP DBNAME=[$userid.]dbname[.copyname][,SHARED-RETRIEVAL]

           [,{n | [n],bufferid}]
```

Default value:

Depends on whether or not PP MAXDB has also been specified:

without PP MAXDB
DBH start with only one database (mono-DB operation)
PP DBNAME=*confname*

with PP MAXDB
DBH start with empty configuration
(multi-DB operation is possible if PP MAXDB > 1).

$*userid*

User identification assigned to the database. $*userid* is mandatory here only if the database is cataloged under a different identification from that used on starting the DBH.

dbname

Name of the database

copy-name

Name of the shadow database
copy-name consists of up to seven characters.

SHARED-RETRIEVAL

The database to be attached can also be accessed by other DBHs using SHARED-RETRIEVAL.
In other words, the application programs of all these DBHs will have read access to the attached database.

> ¡ If you specify all possible parameters when using this load parameter, you should abbreviate the keyword SHARED-RETRIEVAL to SHA.

Default value:

If SHARED-RETRIEVAL is not specified, EXCLUSIVE-UPDATE applies, i.e. the DBH has exclusive access to the database to be attached; application programs have both read and update access.
Default value for a shadow database: SHARED-RETRIEVAL

> ¡ If the DBH has exclusive access to the database to be attached (EXCLUSIVE-UPDATE), but the application programs are only allowed read access, you will also need to specify the DAL command ACCESS RETRIEVAL,DB=dbname

n Size of the user buffer pool for the database in Mbytes. If the parameter *bufferid* is specified, the buffer pool specified by *n* is created as a shared user buffer pool (see below):

n = 1 .. 2047

In the case of a database with an 8-Kbyte page format, at least 3 Mbytes are created.

n = 0 The size of the user buffer pool has already been defined in another database.

n not specified and *bufferid* not specified

The database is buffered in the system buffer pool corresponding to its page format. If this system buffer pool does not exist, activation of the database is rejected.

n not specified and *bufferid* specified

The size of the user buffer pool has already been defined in another database.

> The value *n* must be specified immediately after the comma and must not be separated by blanks (see "DBH load parameters").
>
> *Example*
> `PP DBNAME=dbname,n`

bufferid buffer pool identifier

up to 6 alphanumeric characters (no special characters)
bufferid must be specified if the user buffer pool specified by *n* is to be used as a shared buffer pool for several databases.

The PP DBNAME parameter is used to specify which databases are to be attached to the DB configuration on starting the DBH. If a database that is to be attached is inconsistent, it is made consistent by means of a warm start.

It is also possible to specify whether access to the given database is to be read-only. If the SHARED-RETRIEVAL operand is omitted, both read and update access to the database are allowed.
If desired, you can also specify the size of an exclusive buffer pool for the named database. This buffer pool will be created in addition to the system buffer pool and only be used to buffer pages of the specified database. If you assign an identifier to the buffer pool with the parameter *bufferid*, you can assign it as an exclusive, shared user buffer pool to several databases.

There can be several shared user buffer pools in a single session.

PP DBNAME must be specified **once** for **each** database to be attached. This applies to all databases that are to be attached immediately on starting the DBH. PP DBNAME may be specified a maximum of 222 times.

No two databases within a DB configuration may have the same *dbname*.
If there are any databases not cataloged under the user ID with which the session was started, all realms, ALOG files, and the HASHLIB must be declared shareable (SHARE=YES).

> **CAUTION!**
> The database files must not be write-buffered or write/read buffered on volatile media (see section “Using DAB caching for UDS/SQL”).

The effects of specifying RETRIEVAL or omitting the option are shown in the table below:

Option Effect	No option specified	SHARED-RETRIEVAL
Which usage mode is set when the realms are opened?	UPDATE	RETRIEVAL
If AFIM logging is activated, are the ALOG files opened and written to?	yes	no
Does the DBH allow a warm start to be performed if an inconsistent database is attached?	yes	no
Is READY UPDATE permitted for this database?	yes	no
Can realms be dynamically attached and detached by means of DAL commands?	yes	no
Can multiple DBHs access this database concurrently in SHARED-RETRIEVAL mode?	no	yes

Table 10: Effects of the RETRIEVAL option in PPDBNAME

When a DB is attached while the DBH is being initialized, the compatibility of the ALOG specifications with a UDS /SQL pubset declaration which has already been tested is checked. A DB cannot be attached if one of the ALOG logging specifications which were made using the DEFAULT-SUPPORT or RESERVE-SUPPORT parameter of the BMEND statement START-LOG is outside the pubset space of the current UDS/SQL pubset declaration.

Errors are indicated by one or more of the following messages:

```
UDS0755 UDS USER ERROR: CATID MISSING IN UDS PUBSET DECLARATION (CURRENT):
<catid> , ALOG-DEFAULT ,DB: <dbname> (...,tsn4)
```

or

```
UDS0755 UDS USER ERROR: CATID MISSING IN UDS PUBSET DECLARATION (CURRENT):
<catid> , ALOG-RESERVE ,DB: <dbname> (...,tsn4)
```

The following message is then displayed:

```
UDS0720 UDS ADD ORDER REJECTED: ALOG-SUPPORT NOT WITHIN SCOPE OF UDS PUBSET
DECLARATION (...,tsn4)
```

Message

```
UDS0719 UDS ENFORCES DROP DB=dbname FOR ERROR HANDLING (...,tsn4)
```

enables the error to be assigned to the database. The specifications for `ALOG DEFAULT-SUPPORT` and `RESERVE-SUPPORT` can be displayed using the BMEND statement `SHOW-LOG-INFORMATION`.

This does not hinder the startup of the DBH (without the database affected).

Mono-DB operation

If PP DBNAME and PP MAXDB are not specified, the configuration name specified in the command:

`/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=`*`configuration-name`*

is also the database name of the only database to take part in this session (mono-DB operation).

Please note that the usage mode SHARED-RETRIEVAL is not possible for this database, and that the database then cannot be accessed by several DBHs in parallel (see section “Parallel database operation”).

Mono-DB operation can only be started using the user identification under which the database is cataloged.

Notes on reconfiguration

If the existing configuration is to be altered after a session abort (reconfiguration), the following must be borne in mind:

- A RETRIEVAL option cannot be specified to attach an inconsistent database, since the DBH must access the database in the EXCLUSIVE-UPDATE mode in order to perform a warm start.

  In this case the database must therefore first be attached **without** a RETRIEVAL operand and detached again after a successful warm start. If necessary, the ACCESS command may be issued to prevent this database being accessed by transactions of the current session.
- It is **not** possible to attach a number of inconsistent databases to **different** DBHs **simultaneously** if all these databases have become inconsistent in one and the same session. The reason for this is that the warm starts must be performed sequentially because the DBH has in each case to access the requisite RLOG file in EXCLUSIVE-UPDATE mode.
- If a database has become inconsistent under the configuration name A, it cannot be attached to a configuration with the configuration name B by means of a warm start unless there is currently no session running under the configuration name A.

  The reason for this is that there is only one DB status file per configuration, and that is accessed in the EXCLUSIVE-UPDATE mode both in the current session and in the event of a warm start.

Deactivating server tasks due to computer workload (DEACT)

Independent DBH only!

```
PP DEACT={YES | NO}
```

Default value:

YES

YES Specifies that a server task can be deactivated, i.e. that its resources can be deallocated.

Deactivation can occur in the following cases:

- computer workload is too heavy (CPU, memory, paging rate)
- server task task is too long in wait state (e.g. after “PASS“, “LOCK“, “MESSAGE“)
- server task requires too many system resources (e.g. CPU)

NO Specifies that a server task cannot be deactivated. This causes the server tasks to run more efficiently, since they are not affected by other tasks in the system even in situations when there is a high load on the system, when extensive system resources are consumed, and in wait states.

Other applications could, however, be adversely affected as a result.

The load parameter PP DEACT can be used to control how server tasks are to be handled by the operating system in high-load situations. See also section “Load parameters that affect processor utilization”.

For this parameter to take effect, an ADD-USER command with the INHIBIT DEACTIVATION=YES operand for the user ID in which the DBH was started is a prerequisite (see the commands manuals for "BS2000 OSD/BC").

When using the DEACT parameter it is important to have precise knowledge of the system load. Only when the parameter is used correctly can the overall situation be improved.

Resolving interconfiguration deadlocks or deadlocks involving openUTM (DEADTIME)

Independent DBH only!

```
PP DEADTIME=n
```

Default value:

60

n Time in seconds. An interconfiguration deadlock or a deadlock involving openUTM will be resolved *n* to *1,5n* seconds after it develops.

n = 5..900

If PP DEADTIME is made too small, transactions may be rolled back unnecessarily, for example with IQS or openUTM multistep transactions.

Specifying the minimum size for the distribution pool (DIP-SIZE)

For UDS-D

```
PP DIP-SIZE=n
```

Minimum value:

1024 Kbytes if PP MPSEG=STD is specified;
64 Kbytes if PP MPSEG=64K is specified.

n Size of distribution pool in Kbytes

n = 1..16384

n is rounded up to the nearest integral multiple of 1024 for PP MPSEG=STD and up to the nearest integral multiple of 64 for PP MPSEG=64K. For PP MPSEG=64K, values above 8192 are not advisable.

If *n* > 16384, the DBH uses 16384 Kbytes.

If the storage space requirement calculated by the DBH exceeds the minimum or the specified value, the value computed by the DBH is used when creating the distribution pool.

If PP MPSEG=STD is specified, the distribution pool is created in the upper address space of the UDS/SQL tasks (> 16 MB). In user tasks the distribution pool is created in the lower or upper address space depending on AMODE (24/31). If PP MPSEG=64K is specified, the distribution pool is located in the lower address space in all tasks. If applications in AMODE 24 are also used in the session, the size of the distribution pool must be set such that it fits into the lower address space of this application.

The size of the actually created distribution pool can be obtained with the load parameter PP PARLIST or from the DIP-SIZE value output by the DAL command DISPLAY PP.

Defining the number of remote databases (DISDB)

For UDS-D

```
PP DISDB=n
```

Default value:

1

n Defines the maximum number of remote databases which can be accessed per transaction. Thus no more than *n* remote processing chains can run per transaction.

n = 1..32

PP DISDB is only effective if the DBH load parameter is PP DISTRIBUTION != NO.

> The DBH load parameters PP DISDB, PP MAXDB and PP TRANSACTION affect the size of the communication pool and the distribution pool. It is not advisable to select the maximum values for all of these parameters, since this could make the memory pools too large and potentially create a memory bottleneck below 16 Mbytes, depending on the address space situation.

Defining the input file for the distribution table (DISTABLE)

For UDS-D

```
PP DISTABLE=[:catid:][$userid.]file-name
```

:*catid*:

BS2000 catalog identifier (see section “Using pubsets in UDS/SQL”)

$*userid*

User identification to which *file-name* is assigned

file-name

Name of the input file for the distribution table.

UDS-D reads the input file and from it generates the distribution table (see section “Structure of the input file for the distribution table”). The application program’s only access to remote databases is via the subschemas listed in the distribution table.

If the DBH load parameter PP DISTABLE is omitted, remote application programs can still access the local configuration, but local application programs cannot access remote configurations.

PP DISTABLE is only effective if the DBH load parameter is PP DISTRIBUTION != NO.

Controlling participation in UDS-D operation (DISTRIBUTION)

For UDS-D

```
PP DISTRIBUTION={NO | STANDBY | START}
```

Default value:

NO

NO In this session, remote application programs cannot access the databases of the local configuration and local applications cannot access the databases of remote configurations. No UDS-D-specific tables are generated.

STANDBY

UDS-D operation is only prepared, i.e. UDS-D-specific tables are generated, but the UDS-D task UDSCT is not started.

In this session participation in UDS-D operation is not possible until UDS-D operation has been started with the DAL command &START DISTRIBUTION.

START

UDS-D operation begins on starting the DBH as a result of initiating the UDS-D task UDSCT.

If you set the DBH load parameter PP DISTRIBUTION != NO, you should also set the DBH load parameter PP LOG != NO (see “Maintaining the RLOG File (LOG)”).
If the DBH load parameter PP LOG = NO, remote application programs will have only read access to the local configuration, and local application programs will likewise have only read access to remote configurations.

Determining the scope of a dump (DUMP)

Independent DBH only!

```
PP DUMP={STD | ALL}
```

Default value:

ALL

STD A reduced-scope dump is generated, depending on the error situation and the BS2000 version used.

ALL A dump of the entire user address space is generated.

The load parameter DUMP=ALL should be used in particular when the dump generated using DUMP=STD is insufficient for diagnosing a reproducible error.

Terminating parameter input (END)

```
PP END
```

PP END terminates the input of DBH load parameters. PP END must always be specified.

Executing I/Os asynchronously or synchronously (IO)

Independent DBH only!

```
PP IO={ASYNC | SYNC}
```

Default value:

ASYNC

ASYNC

Causes server tasks to execute all I/Os asynchronously. This eliminates I/O-related waiting periods for the server tasks and thus prevents bottlenecks in high-load situations.

This setting can, however, result in CPU overconsumption in situations where the load is too low.

SYNC

Causes server tasks to execute all I/Os synchronously, which leads to I/O-related waiting periods. The number of server tasks may need to be increased as a result (see the load parameter PP SERVERTASK in chapter "DBH load parameters").

This setting is therefore recommended only in low-load situations or if a small number of openUTM transactions is involved.

If PP IO=SYNC is specified, the load parameter PP SCHEDULING will have no effect.

Defining the locking protocol (LOCK)

```
PP LOCK={STD | SHARED | EXCLUSIVE}
```

Default value:

STD

STD Once a record has been read (see "Locking granularity"), it continues to be subject to a SHARED lock until the next record is read (see "Lock type"). A record that has been updated is subject to an EXCLUSIVE lock until the end of the transaction.
A lock set using the DML statement KEEP during RETRIEVAL processing is of the SHARED type, and a lock set during UPDATE processing is of the EXCLUSIVE type. These locks remain set until the DML statement FREE is issued.

SHARED

As for STD, with one exception:

A lock set using the DML statement KEEP is always of the SHARED type.

EXCLUSIVE

A record read during UPDATE processing is subject to an EXCLUSIVE lock until the end of the transaction.

Locking granularity:

The unit to be locked is always the entire page that contains the record involved.

Lock type:

SHARED
protects against updating by another transaction

EXCLUSIVE
protects against reading and updating by another transaction

Processing type:

Provided that updates are permitted in principle for a realm (database is attached with UPDATE, realms or databases have not been deactivated explicitly (DAL) or implicitly (error handling) or locked and no realm locks have been set using BPRIVACY), the processing type is always UPDATE if SQL is used. If SQL is not used, it is UPDATE only if READY-USAGE-MODE=UPDATE is set.

The parameter applies to all databases participating in the session.

You should specify PP LOCK=SHARED if you use the DML statement KEEP in order to be able to read data repeatedly, but not for the purpose of serialization with respect to other transactions.

By reducing parallel processing, PP LOCK=EXCLUSIVE can be used to achieve shorter path lengths and less deadlocks. You should specify PP LOCK=EXCLUSIVE only if the following points apply to all UPDATE transactions:

- All records that are read are also updated.

- There are no searches (FIND-7) running sequentially.
- Before each access to the member record of a nonsingular set, the owner record is read and modified.

If one of these conditions is not met, concurrency may be diminished or it may not be possible to decrease the path length as anticipated. In this case, you should carry out a performance comparison to check whether it is worthwhile to use PP LOCK=EXCLUSIVE.

Maintaining the RLOG File (LOG)

```
PP LOG={NO |
        :catid: |
        PUBLIC
        (priv-vsn-1/device-1[,priv-vsn-2/device-2[,priv-vsn-3/device-3]])

        (vsn-1[,vsn-2[,vsn-3]])}
```

Default value:

There is no default value.
PP LOG must be entered, or the DBH start will be aborted.

NO The DBH does not maintain an RLOG file.
When LOG=NO is specified

- the independent DBH does not allow updating transactions
- the linked-in DBH does allow updating transactions
- with distributed processing, the independent DBH does not allow any UPDATE transactions distributed via UDS-D (RETRIEVAL transactions are permitted).

:*catid*: BS2000 catalog ID (see section “Using pubsets in UDS/SQL”).

PUBLIC

The RLOG file is created on public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID.

priv-vsn

The RLOG file is created on private disk. Up to three volume serial numbers can be specified, no two of which may be the same. The volume serial number may comprise up to six alphanumeric characters.

device Device type of the private disk. The device type may comprise up to eight alphanumeric characters.

vsn The RLOG file is created on disks which are allocated to an SF pubset or a volume set of an SM pubset. Up to three VSNs can be specified, no two of which may be the same.
The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). Please also take into account “Additional conditions for VSN specifications” .

The load parameter PP LOG determines whether or not the DBH is to create an RLOG file for the session to be started.

The volumes for the original files are specified. Even when a duplicate RLOG file is to be maintained, the specification made with PP LOG determines only where the original files will be kept.
If a duplicate RLOG file is to be maintained, the volumes for the duplicate files must be specified with the load parameter PP LOG-2.
Whether or not a duplicate RLOG file is to be maintained is specified on starting the DBH. This determination cannot be changed later with the DAL command MODIFY. The DAL command MODIFY can be used to change the volumes for the RLOG files.

The amount of storage for the RLOG files is specified with the load parameter PP LOG-SIZE.

Reserve volumes for single and duplicate RLOG files can be specified with the load parameter PP RESERVE.

While the DBH is being initialized, a check is made to see whether the PP LOG specifications are compatible with a UDS/SQL pubset declaration which has already been checked.

In the event of an error, the load parameter concerned is output and the following message is issued:

```
UDS0711 UDS PROGRAM PARAMETER REJECTED: CATID NOT WITHIN SCOPE OF UDS PUBSET
DECLARATION (...,tsn4)
```

DBH startup is then aborted after the load parameter check has been completed.

Single RLOG file

The DBH maintains **one** RLOG file (original) per configuration. This RLOG file always consists of two physical original files.

File names when a single RLOG file is maintained

confname.RLOG.*rlog-time-stamp*.1

confname.RLOG.*rlog-time-stamp*.2

confname specifies the configuration name.

rlog-time-stamp

specifies the time at which DBH starts to use the RLOG file.
Both copies of the RLOG file are given the same time stamp.
rlog-time-stamp is formed as follows:
yymmddhhmmss

1,2:
differentiate between the two cyclically used, physical copies of the RLOG file.

Duplicate RLOG files

The DBH maintains two RLOG files (original and duplicate) per configuration.

PP LOG specifies the volumes for the two original files and PP LOG-2 specifies the volumes for the two duplicate files.

PP LOG and PP LOG-2 may be given in any order when specifying the load parameters.

A duplicate RLOG file can be maintained only if the original and duplicate files are maintained on different volumes. At initialization, UDS/SQL checks whether the duplicate file resides on a different volume. If it is evident from the syntax analysis of the DBH load parameter that the volumes are the same, the DBH start is aborted. Otherwise, i.e. if the identical volumes are noted only on accessing the volume, the databases involved in the session are protected by an UPDATE lock.

File names when a duplicate RLOG file is maintained

The DBH maintains **two** RLOG files, i.e. an original and a duplicate (SAVE).

The following four physical files are created:

confname.RLOG.*rlog-time-stamp*.1

confname.RLOG.*rlog-time-stamp*.2

confname.RLOG.*rlog-time-stamp*.1.SAVE

confname.RLOG.*rlog-time-stamp*.2.SAVE

In the event of an error in one copy of the RLOG file, a rollback is still possible despite the error, using the duplicate; this enables the databases to be kept consistent.

¡ When LOG=NO is specified, the linked-in DBH allows update transactions. If an error then occurs during the session and makes the rollback of a transaction or a warm start necessary, those databases of the DB configuration which have been updated in the course of the session are rendered irrecoverably **inconsistent**. In such a case you would have to use a backup of the databases and update the inconsistent databases up to the last consistency point using the BMEND utility routine (UPDATE-DATABASE, DEADLINE=STD statement).
LOG=NO should therefore only be used to load large masses of data quickly into databases after the data resource has been saved.

! CAUTION!

The RLOG files must not be write-buffered or write/read buffered on volatile media (see section “Using DAB caching for UDS/SQL”).

Additional conditions for VSN specifications

The following special rules apply for VSN specifications when the RLOG files are created on disks which are allocated to an SF pubset or a volume set of an SM pubset:

- VSN specifications are permissible only when authorization exists for physical allocation of public storage space on the pubset concerned.

- In both PUB notation (PUBpxx) and dot notation (pp[pp].[xy]z) it is permissible to omit the sequence number of the disk (xx bzw. [xy]z), in which case only a single VSN specification is possible.

  This enables the RLOG file to be created on any volumes of an SM pubset's volume set.

  Examples:

(PUBX01, P UBX02)	RLOG file on volumes PUBX01 and PUBX02 in the singlecharacter SF pubset X or volume set X of an SM pubset
(PUBX)	RLOG file on any volume in the single-character SF pubset X or volume set X of an SM pubset
(ABC.01,ABC. 02)	RLOG file on volumes ABC.01 and ABC.02 in the threecharacter SF pubset ABC or volume set ABC of an SM pubset
(ABC.)	RLOG file on any volume in the three-character SF pubset ABC or volume set ABC of an SM pubset

- If multiple volumes are specified for an RLOG file, these must belong to the same volume set, i.e. the VSNs must contain the same catalog ID. A file cannot extend over multiple volumes sets.
- Twin logging files must be created on different volume sets because when a volume fails the entire volume set is generally no longer available.
- In addition, one of the twin logging files may also not be created on a volume set of an SM pubset if the other is created in this SM pubset on account of a :catid: or PUBLIC specification.
- The following must be borne in mind when the control volume set of an SM pubset is specified for a copy of RLOG or two volume sets of an SM pubset are specified for the RLOG original and duplicate:

  If the control volume set of an SM pubset fails, the entire SM pubset is temporarily no longer accessible. (Reconstruction of the SM pubset is possible.)

Maintaining the duplicate RLOG file (LOG-2)

```
PP LOG-2={:catid: |
         PUBLIC
         (priv-vsn-1/device-1[,priv-vsn-2/device-2[,priv-vsn-3/device-3]])

         (vsn-1[,vsn-2[,vsn-3]])}
```

:*catid*:

BS2000 catalog ID (see section “Using pubsets in UDS/SQL”).

PUBLIC

The duplicate RLOG file is created on public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID.

priv-vsn

The duplicate RLOG file is created on private disk. Up to three volume serial numbers can be specified, no two of which may be the same. The volume serial number may comprise up to six alphanumeric characters.

device Device type of the private disk. The device type may comprise up to eight alphanumeric characters.

vsn The RLOG file is created on disks which are allocated to an SF pubset or a volume set of an SM pubset. Up to three VSNs can be specified, no two of which may be the same.

The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). Please also take into account “Additional conditions for VSN specifications”.

If you specify the load parameter PP LOG-2 then two copies of the RLOG file are written The load parameter value that you specify determines the data media for the duplicate files.

The volumes for the original files are specified with the load parameter PP LOG. PP LOG and PP LOG-2 may be given in any order.

Whether or not a duplicate RLOG file is to be maintained is specified on starting the DBH. This determination cannot be changed later with the DAL command MODIFY. The DAL command MODIFY can be used to change the volumes for the RLOG files.

A duplicate RLOG file can be maintained only if the original file and duplicate file are kept on different volumes. At initialization, UDS/SQL checks whether the duplicate file resides on a different volume. If the volumes are the same, the DBH start is aborted.

Please also take into account the information on "Duplicate RLOG files" in chapter "DBH load parameters".

The amount of storage for the RLOG files is specified with the load parameter PP LOG-SIZE.

Reserve volumes for single and duplicate RLOG files can be specified with the load parameter PP RESERVE.

While the DBH is being initialized, a check is made to see whether the PP LOG-2 specifications are compatible with a UDS/SQL pubset declaration which has already been checked.

In the event of an error, the load parameter concerned is output and the following message is issued:

```
UDS0711 UDS PROGRAM PARAMETER REJECTED: CATID NOT WITHIN SCOPE OF UDS PUBSET
DECLARATION (...,tsn4)
```

DBH startup is then aborted after the load parameter check has been completed.

Specifying the amount of storage in RLOG files (LOG-SIZE)

```
PP LOG-SIZE=([primary][,[secondary]])
```

Default value:

192 PAM pages each (2048 bytes/page)

primary Primary allocation in PAM pages
primary = 1..9999999

secondary

Secondary allocation in PAM pages
secondary = 1..32767

The load parameter PP LOG-SIZE is used to specify the following:

- The primary allocation, i.e. the amount of storage space to be initially allocated to each RLOG file.
- The secondary allocation, i.e. the amount of storage space to be automatically allocated when additional space is needed during file processing.

If for the secondary allocation you specify a value smaller than the default value of 192, the value specified will be rounded up to 192.

The primary allocation should correspond to the anticipated size of the file to be created.

For large files, the values chosen for the primary and secondary pages should be multiples of the management units of 24 or 192 PAM pages.

Defining the number of databases in the DB configuration (MAXDB)

```
PP MAXDB=n
```

Default value:

Number of databases specified in the load parameter PP DBNAME. If DBNAME has not been specified, the DBH automatically goes into the mono-DB mode and assumes a value of 1.

n Maximum number of local databases participating in the session
n = 1..222

> ¡ In addition to the 222 local databases, up to 32 remote databases can be accessed. In other words, a total of 254 databases may be involved in a session.

The load parameter PP MAXDB sets a limit on the number of databases that can be attached at any one time to the session to be started.

If dynamic attachment of databases (ADD DB) causes the limit specified in the MAXDB parameter to be exceeded, the DBH rejects attachment of the database.

The DBH makes the value of *n* equal to the number of PP DBNAME load parameters if *n* is smaller than the number of databases specified.

PP MAXDB **must be specified** for a session in which databases are attached dynamically, i.e. where the total number of attached databases may exceed the number at start of the session.

PP MAXDB **need not be specified** for a session in which no databases are attached dynamically.

> ¡ The DBH load parameters PP DISDB, PP MAXDB and PP TRANSACTION affect the size of the communication pool and the distribution pool. It is not advisable to select the maximum values for all of these parameters, since this could make the memory pools too large and potentially create a memory bottleneck below 16 Mbytes, depending on the address space situation.

Defining the segment size of memory pools (MPSEG)

Independent DBH only!

```
PP MPSEG={STD | 64K}
```

Default value:

STD

STD The memory pools are created in the UDS/SQL tasks in the upper address space (> 16 Mbytes) with a segment size of 1 Mbyte.
Application programs in AMODE 31 connect to this pool such that they also lie above 16 Mbytes in the upper address space.
With application programs in AMODE 24, these pools lie below 16 Mbytes in their virtual address space.

64K The communication pool, transfer pool, distribution pool and SSITAB pool are created below 16 Mbytes with a segment size of 64 Kbytes.
The remaining memory pools lie above 16 Mbytes and are created with a segment size of 1 Mbyte.

This operand should be specified only if the segment size of 1 Mbyte results in memory bottlenecks, since the segment size of 64 Kbytes can have a negative effect on performance.

Controlling system behavior during collisions between pending requests and UPDATE transactions (ORDER-DBSTATUS)

Independent DBH only!

```
PP ORDER-DBSTATUS={STD | SPECIAL}
```

Default value:

STD

STD New UPDATE transactions or processing chains that collide with the execution of existing CHECKPOINT or NEW RLOG requests are rejected with the global status code 12122, and the transaction is rolled back with CANCEL.

SPECIAL

New UPDATE transactions or processing chains that collide with the execution of existing CHECKPOINT or NEW RLOG requests are rejected with the specific status code 12124, and the transaction is rolled back with CANCEL.

The load parameter PP ORDER-DBSTATUS can be used to control system behavior in cases where the DBH has been instructed by a PERFORM command or a system-internal PERFORM (e.g. because of an error situation) to execute pending requests (CHECKPOINT, NEW RLOG) and is therefore rejecting new UPDATE transactions or processing chains. As a result, the specification ORDER-DBSTATUS=SPECIAL makes it possible to react specifically to certain wait situations in the application program.

Listing the load parameters used (PARLIST)

```
PP PARLIST={YES | NO}
```

Default value:

NO

YES Load parameters are listed

NO Load parameters are not listed

The load parameter PP PARLIST can be used to list the current load parameters on the database administrator’s display terminal or on SYSOUT when starting the DBH or restarting a session.

Defining passwords for files used by the DBH (PASSWORD)

```
PP PASSWORD={NONE | STANDARD | password( password, password,...)}
```

Default value:

STANDARD

NONE

UDS/SQL requires only the password specified in PP CATPASS.

STANDARD

UDS/SQL requires only the password specified in PP CATPASS and the standard password C'UDS' BLANK''.

password
(*password*,*password*,...)

In addition to the password specified in PP CATPASS, UDS/SQL requires further passwords. Via *password* the database administrator transfers the password(s) required for the session. Up to 100 passwords can be specified, distributed over several PP PASSWORD specifications.

password may be up to four bytes in length and is represented as follows:

C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

d:
where *d* is a decimal number (comprising up to eight digits and a sign) to be converted to a binary value

The syntax conventions for BS2000 apply (see ADD-PASSWORD in the commands manual for "BS2000 OSD/BC").

The PP PASSWORD parameter is used to specify one or more passwords for files which already exist when they are to be opened by the DBH, such as user realms and DBDIR as well as any ALOG files created beforehand, the DB status file and temporary realms (see the section “Assigning passwords to UDS/SQL files”).

Password protection of the file containing the DBH load parameters is the responsibility of the database administrator. The set of passwords can be changed dynamically with ADD PW (see "Attaching databases, realms and passwords (ADD)") and DROP PW (see "Detaching databases, realms and passwords (DROP)").PP PASSWORD may be specified more than once.

The password C'UDS'BLANK'' and read passwords specified via PP CATPASS, PP PASSWORD or the DAL command ADD PW should not be used for other files, because UDS/SQL issues these passwords internally, so in special situations files would become accessible which users want to protect.

Responses in error situations

As the DBH load parameters are usually recorded in a cataloged file, incorrect PP PASSWORD specifications cannot be corrected by simple repetition during the UDS/SQL start phase. The response to the following input errors is always to abort following PP END:

- syntax error in PP PASSWORD
- PP PASSWORD=NONE/STANDARD specified more than once
- PP PASSWORD truncated when read in
- more than 100 passwords specified with PP PASSWORD

Controlling handling of privacy checks (PRIVACY-CHECK)

```
PP PRIVACY-CHECK={STD | NO-KSET | OFF}
```

Default value:

STD

STD The privacy check is performed.

In order to perform the privacy check, the user group name must be in the format described under the section on BPRIVACY in the "Creation and Restructuring" manual.

NO-KSET

The privacy check is performed.

The KSET name in the user group name is not checked in the course of a privacy check for openUTM.

OFF No privacy check is performed.

Handling transactions in the PTC state in the event of errors (PTCSYNCH)

```
PP PTCSYNCH=([{WAIT | ABORT | COMMIT}][,[{WAIT | ABORT | COMMIT}]])
```

Default value:

(WAIT,WAIT)

For UDS-D/openUTM

The **first** value of the PTCSYNCH parameter applies to secondary subtransactions or openUTM transactions in the PTC condition at the time of a **warm start**.

For UDS-D

The **second** value of the PTCSYNCH parameter applies to secondary subtransactions which are in the PTC condition **in the course of the session** and for which the condition of the corresponding primary subtransaction cannot be ascertained.

WAIT The secondary subtransactions/openUTM transactions wait until they are terminated either by a message from the corresponding primary subtransaction from openUTM or explicitly by the database administrator using the DAL command ABORT, OPTION=PTC or COMMIT.

ABORT

UDS/SQL rolls back the secondary subtransactions/openUTM transactions and issues a warning.

COMMIT

UDS/SQL terminates the secondary subtransactions/openUTM transactions, commits the updates and issues a warning to the database administrator.

With the DBH load parameter PP PTCSYNCH you may control whether secondary subtransactions or openUTM transactions in the PTC state should terminate with FINISH or with FINISH WITH CANCEL after a time interval defined by PP CHCKTIME.

The values of the DBH load parameter PP PTCSYNCH can be dynamically changed by the database administrator during the session with the aid of the DAL command MODIFY PTCSYNCH.

> ¡ If the DBH load parameter PP PTCSYNCH is not set to (WAIT,WAIT), the interconfiguration consistency (or UDS/SQL openUTM consistency) is at risk (see section “Terminating the PTC state”).

Specifying reserve volumes for the RLOG files (RESERVE)

```
PP RESERVE=
   {NONE |
  :catid: |
   PUBLIC
   (priv-vsn-1/device-1[,priv-vsn-2/device-2[,priv-vsn-3/device-3]])

   (vsn-1[,vsn-2[,vsn-3]])}
```

Default value:

NONE

No reserve volumes are specified.

:*catid*:

BS2000 catalog ID (see section “Using pubsets in UDS/SQL”).

PUBLIC

Public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID.

priv-vsn

The RLOG file is created on the private disks specified as reserve volumes. Up to three volume serial numbers can be specified, no two of which may be the same. The volume serial number may comprise up to six alphanumeric characters.

device Device type of the private disk. The device type may comprise up to eight alphanumeric characters.

vsn Disks which are allocated to an SF pubset or a volume set of an SM pubset are specified as reserve volumes for the RLOG file. Up to three VSNs can be specified, no two of which may be the same. The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). Please also take into account “Additional conditions for VSN specifications”.

The load parameter PP RESERVE is used to assign the reserve volumes for the volumes specified with the load parameters PP LOG and PP LOG-2, except when no reserve volumes are specified.

If a volume specified with the initial allocation PP LOG or PP LOG-2 is unavailable when an RLOG file is created, the reserve volume from PP RESERVE is used in its place. For reserve volumes you must specify different volumes from those specified in the initial allocation. This is not checked when the DBH load parameters are read in, but instead only when a reserve volume is actually needed.

If the volumes of the reserve allocation PP RESERVE are also unavailable, an update lock is placed on the entire configuration. The update lock remains in effect until the volumes become available or another volume is allocated, and the RLOG file is activated with the DAL commands NEW LOG and PERFORM.

Switching to reserve volumes is limited to cases of error; at the earliest opportunity the DBH will attempt again to set up the files involved as specified in the initial allocation.

There is just one specification of reserve file allocation per configuration.

While the DBH is being initialized, a check is made to see whether the PP RESERVE specifications are compatible with a UDS/SQL pubset declaration which has already been checked.

In the event of an error, the load parameter concerned is output and the following message is issued:

```
UDS0711 UDS PROGRAM PARAMETER REJECTED: CATID NOT WITHIN SCOPE OF UDS PUBSET
DECLARATION (...,tsn4)
```

DBH startup is then aborted after the load parameter check has been completed.

Grouping together request results for user tasks (RESULT-DELAY)

Independent DBH only!

```
PP RESULT-DELAY=n
```

Default value:

0

n Number of grouped request results for user tasks.

n = 1..*m*

m = PP TRANSACTION

The load parameter PP RESULT-DELAY permits a throughput increase in peak load situations by grouping together the request results for the user tasks. See also section “Load parameters that affect processor utilization”.

You should specify this parameter only under the following conditions:

- You are using a multiprocessor system with more than three processors.
- The number of generated server tasks is the same as the number of processors that can be utilized by the server tasks. This usually means that the server tasks must have a high priority.

If PP SCHEDULING=ASYMMETRIC is set at the same time, the number of request results actually grouped may be restricted.

> If the value that you specify is higher than the one specified for PP TRANSACTION, it is adjusted to the value of PP TRANSACTION without warning.

Controlling scheduling behavior in high-load situations (SCHEDULING)

Independent DBH only!

```
PP SCHEDULING={SYMMETRIC | ASYMMETRIC}
```

Default value:

SYMMETRIC

SYMMETRIC

All server tasks have equal precedence when processing pending DML requests. This setting is favorable if the server tasks are well-loaded or if no high BS2000 priority can be assigned to them.

ASYMMETRIC

An attempt is made to distribute the DML requests across all server tasks with the objective of achieving a 100% load on *n*-1 server tasks in a high-load situation if possible. This results in better processor utilization, increased throughput, and a more favorable multiprocessor factor.

This setting is only meaningful for multiprocessor systems and for configurations with more than one server task. In addition, appropriate BS2000 task priorities must be set for the server tasks (see chapter “Optimizing performance”).

Asymmetric processing could result in slower response times for individual transactions in overload situations. This is, however, usually offset by the higher overall throughput and the quicker resolution of the overload situation.

The load parameter PP SCHEDULING can be used to control the internal behavior of multiple server tasks in cases where there are strong fluctuations in relatively high-load situations. See also section “Load parameters that affect processor utilization”.

> ¡ If PP SERVERTASK=1 or PP IO=SYNC is specified, the load parameter PP SCHEDULING will have no effect.

Defining the number of server tasks (SERVERTASK)

Independent DBH only!

```
PP  SERVERTASK=n
```

Default value:

1

n Number of server tasks to be started

n = 1..30

On starting the DBH, the DBH master task evaluates the load parameter PP SERVERTASK to determine how many server tasks are to be started (see section “How the independent DBH works”).

If all inputs and outputs are handled asynchronously (see the load parameter PP IO), it is sufficient to start one server task on a monoprocessor and a maximum of one server task per processor on a multiprocessor system. The server tasks must then have a correspondingly high priority (see “RUN-PRIO” on "Starting DBH").

See also chapter “Optimizing performance”.

ℹ If PP SERVERTASK=1 is set, the load parameter PP SCHEDULING will have no effect.

Specifying the size of the SSITAB pool (SIP-SIZE)

Independent DBH only!

```
PP SIP-SIZE=n
```

Default value:

1024 Kbytes if PP MPSEG=STD is specified;
128 Kbytes if PP MPSEG=64K is specified.

n Size of the SSITAB pool in Kbytes

n = 1..16384

n is rounded up to the nearest integral multiple of 1024 for PP MPSEG=STD and up to the nearest integral multiple of 64 for PP MPSEG=64K. For PP MPSEG=64K, values above 8192 are not advisable.

If *n* > 16384, the DBH uses 16384 Kbytes.

When determining the pool size, you should take the following into account:

- the user address space
- the sum of the lengths of all SSITAB modules or of those to be processed concurrently.

To avoid frequent loading and unloading of the SSITAB modules in the pool, you should specify a size for the pool that is large enough to accommodate all the SSITAB modules used.

If PP MPSEG=STD is specified, the SSITAB pool is created in the upper address space of the UDS/SQL tasks (> 16 MB). In user tasks the SSITAB pool is created in the lower or upper address space depending on AMODE (24 /31). If PP MPSEG=64K is specified, the SSITAB pool is located in the lower address space in all tasks. If applications in AMODE 24 are also used in the session, the size of the SSITAB pool must be set such that it fits into the lower address space of this application.

The size of the SSITAB pool can be obtained with the load parameter PP PARLIST or from the SIP-SIZE value output by the DAL command DISPLAY PP.

Defining the number of simultaneously active SQL conversations (SQL)

Independent DBH only!

```
PP SQL=n
```

Default value:

4 (default value for PP TRANSACTION)

n Maximum number of simultaneously active SQL conversations

n = 0..9999

When a new SQL conversation is to be started but doing so would result in the value *n* being exceeded, the DBH attempts to terminate a conversation that has been inactive for the time period specified with PP SQL-LIMIT. If this cannot be done, the new SQL conversation is rejected with an error message (SQL code: -1710).

When determining the size of PP SQL, you must take into account the fact that SQL conversations occupy internal resources (e.g. memory space) throughout their entire lifetimes. Too many concurrent SQL conversations can thus lead to resource bottlenecks.

PP SQL may also be larger than PP TRANSACTION; this is appropriate in a openUTM environment. PP TRANSACTION determines the maximum number of simultaneously active transactions and the maximum number of user tasks that can be attached. A task that does not execute with openUTM cannot open more than one conversation.

PP SQL=0 can be specified to prevent the use of SQL, even if the DBH was loaded with SQL linkage.

See also chapter “The SQL conversation”.

Setting the time period for inactive SQL conversations (SQL-LIMIT)

Independent DBH only!

```
PP SQL-LIMIT=n
```

Default value:

10

n Minimum amount of time, in minutes, for which UDS/SQL will maintain data for inactive conversations

n = 5..999 minutes

When a new SQL conversation is to be started but doing so would result in the value of PP SQL being exceeded, UDS/SQL attempts to terminate conversations that have already been inactive for longer than the time period specified with SQL-LIMIT.

See also chapter “The SQL conversation”.

Writing checkpoints in ALOG files (STDCKPT)

```
PP STDCKPT={YES | NO}
```

Default value:

NO

YES By default the DBH writes checkpoints in the ALOG files for the entire configuration on starting and ending the DBH and on restarting a session.

NO No checkpoints are written at the above-described events.

The load parameter PP STDCKPT is used by the DBH to write checkpoints in ALOG files for all databases of the configuration.

Checkpoints are written by the DBH

- on starting the DBH
- at a normal DBH end (CLOSE CALLS/CLOSE RUN-UNITS)
- following a successful session restart.

The results of writing a checkpoint in the ALOG file are that:

- the ALOG file as written up to that point is closed
- a UDS/SQL message supplies the user with the log interval end (time stamp) of the ALOG file which has just been closed
- a new ALOG file is opened.

The writing of checkpoints causes ALOG files to be closed. This releases the files, e.g. for updating shadow databases (with the BMEND utility routine). The ALOG files that have been closed can also be stored on tape and then deleted from the disk.

Checkpoints are written simultaneously for the entire DB configuration.
Dynamic database attachment and detachment with the DAL commands ADD and DROP may cause the configuration to change as compared to the original configuration on starting the DBH.

When the linked-in DBH is used, the PP STDCKPT parameter is the database administrator’s only possible means of requesting the writing of checkpoints in ALOG files.

Switching the ALOG file

If the database is kept under the configuration user ID, the DBH creates a new ALOG file once the current ALOG file is completed.

If the database is kept under a different user ID, an ALOG successor file must be available under the other user ID, and the file must be shareable.

If no ALOG successor file exists, the following is effected:

- The old ALOG file is closed in a consistent state.
- The ALOG file sequence number in the DBDIR is incremented by one and a marker is set to flag that the ALOG file identified by the ALOG file sequence number is not yet ready for use.
- The database is locked against updates until the ALOG file identified by the ALOG file sequence number is ready for use.
- A message is issued to indicate that there is no ALOG successor file.

The database administrator can then set up an ALOG file using the CREATE-FILE command. By means of the DAL command CHECKPOINT, the database administrator causes the DBH to initialize the new ALOG file and complete the writing of the checkpoint. The lock against updates is released as soon as the ALOG file is ready for use.

ALOG file overflow

An ALOG file overflow occurs when the ALOG file contains insufficient space to close the ALOG file in a consistent state and switch to a new one. There are two types of overflow:

- soft ALOG file overflow (FILE OVERFLOW)

  The FILE OVERFLOW message indicates that the ALOG file is in danger of overflowing, although there is still enough space to reach a consistency point.

  The database is locked against updating transactions. Active transactions are rolled back or forward (to COMMIT), depending on the amount of space left in the ALOG file.

  Once a consistency point has been reached, the ALOG file is switched.

- hard ALOG file overflow (ALOG OVERFLOW)

  The ALOG OVERFLOW message indicates that the ALOG file is full and there is no space to reach a consistency point. AFIM logging is no longer possible and there would be a gap in logging.
  This may occur particularly in the case of long, active transactions.

  The database is locked against all transactions. The DBH drops the database in an inconsistent state.

  You can ascertain the cause of the file overflow from the DMS error code and take the relevant measures.

  Once you have created enough space to extend the ALOG file, you can attach the database once again by means of a warm start.

  Another ALOG OVERFLOW could occur during a warm start, in which case you can repeat the procedure as many times as needed to reach a consistency point.

Defining the number of subschemas (SUBSCHEMA)

```
PP SUBSCHEMA=n
```

Default value:

1

n Number of concurrently usable subschemas per database

n = 1..100

Guide to value of *n*:
Number of **frequently** referenced subschemas, plus reserve for **sporadically** referenced subschemas.

The subschema information area (SSIA) of the addressed subschema must be resident in main memory in order for a processing chain to be executed. The load parameter SUBSCHEMA defines the maximum number of subschemas per database that may be processed simultaneously during the session.

Selecting too low a value for *n* may degrade performance, since the DBH is then frequently compelled to:

- off-load SSIAs that have already been loaded so that new SSIAs can be read
- or even to CANCEL transactions (with BIB status code 132) because the required SSIAs are not loaded and cannot be read due to a lack of space.

Defining the number of KDBS file identifications (SUBTRANSACTION)

```
PP SUBTRANSACTION=n
```

Default value:

0

n Maximum number of KDBS file IDs permitted for all KDBS users at the same time on one database.

n = 1..254

The load parameter PP SUBTRANSACTION is required **only when KDBS is used**.

Defining usage modes for transactions (TA-ACCESS)

Independent DBH only!

```
PP TA-ACCESS={STD | SHARED}
```

Default value:

STD

STD When a processing chain is opened, all usage modes are permitted. Depending on the usage mode set, the realms to be opened are locked against reading or updating for other transactions.

SHARED

When a processing chain is opened, only the usage modes SHARED RETRIEVAL and SHARED UPDATE are permitted. The realms to be opened are not locked.
Any attempt made to open a processing chain in usage mode PROTECTED or EXCLUSIVE is rejected with status code "092".

You should specify PP TA-ACCESS=SHARED if all transactions are opened in usage mode SHARED. This reduces the path length for READY.

The load parameter PP TA-ACCESS applies to all databases participating in the session.

Defining the maximum number of transactions (TRANSACTION)

```
PP TRANSACTION={n | ([n][,m])}
```

Default value:

n : 4 / *m* : 1

n Maximum number of simultaneously open transactions; at the same time, maximum number of user tasks (including openUTM tasks) that can be linked simultaneously to the DBH.

n = 1..225

For UDS-D

m maximum number of secondary subtransactions which the DBH is able to process simultaneously.

m = 1..*n* and *m* <= *n*

The value for secondary subtransactions is operative in UDS-D mode only.

PP TRANSACTION determines

- the number of transaction channels (mainrefs) in the DBH and thus the maximum number of transactions which the DBH can process concurrently,
- the number of task administration entries of the DBH and thus the maximum number of user tasks that can simultaneously be known to the DBH,
- with UDS-D, the maximum number of secondary subtransactions that this DBH can process simultaneously for remote application programs.

A **timesharing** user task is known to the DBH

- from the start of the UDS/SQL program (in the case of COBOL DML and SQL) or from the first READYC call (in the case of CALL DML)
- until the end of the UDS/SQL program.

In **UDS/SQL openUTM mode**, each openUTM task is known to the DBH

- from the start to the end of the task.

When the linked-in DBH is used, the value of load parameter PP TRANSACTION is permanently set to 1.

The DBH load parameters PP DISDB, PP MAXDB and PP TRANSACTION affect the size of the communication pool and the distribution pool. It is not advisable to select the maximum values for all of these parameters, since this could make the memory pools too large and potentially create a memory bottleneck below 16 Mbytes, depending on the address space situation.
It is recommended that you select the value for the maximum number of secondary subtransactions slightly higher than the expected number of secondary subtransactions required. This is because if, for example, the primary subtransaction is rolled back asynchronously (e.g. because of deadlocks or by the administrator), the configurations of the secondary subtransactions are not informed of this immediately, but instead only in the context of the transaction monitoring (see section “Time-controlled monitoring of connections and transactions”). In such cases, the resources of the secondary subtransactions still remain occupied after the primary subtransaction is rolled back, so that they cannot be used by new secondary subtransactions in the time between the rollback of the primary subtransaction and the start of the transaction monitoring. If a secondary subtransaction cannot be opened due to a resource bottleneck, the application program is given the status code 122.

Defining the console for DCAM administration logging and the format of UDS /SQL messages (UCON)

Independent DBH only!

```
PP UCON=C'{(mn) | <x | nnnn}'[,{MSG |UDS}]
```

Default value:

C’<U’,MSG
Logging is output at the local console. If logging cannot take place at the output location set elsewhere, the default value is once again assumed.

Output location:

mn Two-character mnemonic device name.

<x *x* = one-character routing code.

The character ’<’ must be specified.

nnnn Four-character name of the authorized user program with operator functions.

Format:

MSG The messages are logged without a header (default value).

This format should be used if messages from different systems are to be evaluated by one console in a uniform way (e.g. via OMNIS-PROP).

UDS The messages are logged with the formatted, UDS/SQL-specific header. The header is added to messages administered via DCAM if administration via DCAM does not take place directly from a display terminal. For information on the structure and contents of the header, see the "Messages" manual.

The load parameter PP UCON defines the console (UCON) at which DCAM administration logging is to take place and the format of UDS/SQL messages (see "PP ADM" in chapter "DBH load parameters", section “Administration of UDS/SQL via DCAM”, and the manual "Introduction to System Administration").

ℹ Messages sent to the local display terminal (SYSOUT) are always output without a header.

Setting the wait mode of server tasks (WAIT)

Independent DBH only!

```
PP WAIT={EVENT | BUSY}
```

Default value:

EVENT

EVENT

The last active server task waits for an event message before releasing the processor, which can then be used to perform other functions.

BUSY The server task waits actively for the next request. The processor is not released.

The load parameter PP WAIT is used in UDS/SQL peak-load applications to control the way in which the last active server task is to wait for new requests in short-term underload situations. See also section “Load parameters that affect processor utilization”.

You should specify PP WAIT=BUSY only under the following conditions:

- You are using a multiprocessor system with more than three processors.
- The load is consistently so high throughout the entire session that at least one server task always has requests to process.
- Underload situations are rare.

Determining the duration of a warm start (WARMSTART)

```
PP WARMSTART={STD | FAST | VERY-FAST}
```

Default value:

STD

During operation of the independent DBH, the load parameter WARMSTART controls when completed transactions finally enter the database.

If the default value is used, the completed transactions are first written to the RLOG file for reasons of fail-safety. They are not written to the database until the appropriately modified pages are off-loaded from the system buffer pools. This cuts down on the number of unnecessary read and write operations during operation. But in the event of a warm start, the rolling forward of these transactions, which are completed but not yet in the database, must be ensured.

With the values FAST and VERY-FAST, the independent DBH rolls forward the completed transactions even if the appropriately modified pages have not yet been off-loaded from the system buffer pools. This increases the number of write operations to the database during operation, but also means that any warm start that becomes necessary will be completed more quickly because there are fewer completed transactions to be updated. The part of the RLOG file that must be read during the warm start is used as the criterion for the premature addition of completed transactions to the database.

During a warm start, however, not only the parts of the RLOG file that contain completed transactions must be read, but also parts containing open transactions. If, as a result of long transactions, the part containing the open transactions grows large, this limits the effect of the load parameter WARMSTART.

STD
: Completed transactions are not entered in the database ahead of time. The duration of a future warm start is not reduced.

FAST
: Completed transactions are definitively entered in the database before any offloading from the system buffer pools. The part of the RLOG file that must be read during a warm start is reduced by up to 90% compared with STD. The duration of a future warm start is shortened, possibly at the expense of performance during normal operation.

VERY-FAST
: Completed transactions are definitively entered in the database before any offloading from the system buffer pools. The part of the RLOG file that must be read during a warm start is reduced by up to 99% compared with STD. The duration of a future warm start is thus shortened, possibly at the expense of performance during normal operation.

4 Administration

There are three interfaces for the administration of a UDS/SQL session:

- administration via UDSADM
  (see section “Administration of UDS/SQL via UDSADM”)
- administration via DCAM
  (see section “Administration of UDS/SQL via DCAM”)
- administration via INFORM-PROGRAM or SEND-MESSAGE
  (see section “Administration of UDS/SQL via INFORM-PROGRAM”)

You can find a description of the commands of the database administration language DAL as of "The Database Administrator Language DAL".

4.1 Administration of UDS/SQL via UDSADM

The UDSADM program can be used for the administration of a UDS/SQL configuration if the load parameter PP ADM=REMOTE is specified on starting the DBH. Connection requests from the UDSADM program are rejected as long as there is already an administration program or a data display terminal connected to the configuration via DCAM.

UDSADM communicates with the master task and connects itself to the DCAM applications opened by the master task.

UDSADM enables UDS/SQL messages to be received by UDS/SQL and DAL commands to be sent to UDS/SQL.

UDSADM can be linked to only one UDS/SQL configuration at any one time.

Although all database administrators have the same rights, **one** UDS/SQL configuration can have only **one** database administrator at any one time.

UDSADM can run both in batch and interactive mode.

If you encapsulate UDSADM in a procedure, you can have all actions logged by means of the command ASSIGN-SYSLST TO= ... .

Starting UDSADM

UDSADM is started with:
`/START-UDS-ADM`

For reasons of compatibility, the following alias names are also available:
`/START-UDS-ADMINISTRATION`
or
`/UDSADM`

The BS2000 commands described here assume that USD/SQL was installed via IMON. For a more detailed description of the prerequisites, refer to the section "START commands of the UDS/SQL programs" in the manual "Creation and Restructuring".

4.1.1 UDSADM statements

The statement formats of the UDSADM administration program comply with SDF (System Dialog Facility) conventions; see the manuals "SDF Dialog Interface" and the commands manuals of "BS2000 OSD/BC". Uppercase letters printed in boldface denote guaranteed abbreviations of keywords.

Overview of UDSADM statements

Statement	Meaning
CONNECT**-C**ONFIGURATION **C**ONFIGURATION**-N**AME = <name> ,**PAS**SWORD = ***NONE** / <x-string> <c-string> ,**HOST** = ***LOC**AL / <name>	Sets up connection to UDS/SQL configuration
DISCONNECT**-C**ONFIGURATION	Clears connection to UDS/SQL configuration
END	Terminates UDSADM
EXECUTE**-D**AL**-C**MD **CMD** = <dal-cmd>	Executes DAL commands
HELP**-D**AL**-C**MD **?**	Help function for DAL commands
MODIFY**-M**SG**-F**ORMAT **HEADER** = **NO** / **YES**	Sets message output format
MODIFY**-M**SG**-W**AIT**-T**IME **TIME** = **STD** / <integer 1..7200 *seconds*>	Sets wait time for DAL commands
SET**-R**ECEIVE**-M**ODE	Activates permanent receive task
SHOW**-C**ONNECTION**-A**TTRIBUTES	Displays information on connection
SHOW**-O**UTSTANDING**-M**SG	Displays outstanding messages

Table 11: UDSADM statements

UDSADM statements entered incorrectly are rejected with a specific message and can be corrected. Each UDSADM statement entered correctly is executed immediately.

Statement selection rules

Statement	Selection rules
END HELP-DAL-CMD MODIFY-MSG-FORMAT MODIFY-MSG-WAIT-TIME	These statements may be entered at any time.
DISCONNECT-CONFIGURATION	This statement is permitted only for a connection that was previously set up using CONNECT-CONFIGURATION.
CONNECT-CONFIGURATION	This statement is permitted for a connection not yet set up.
EXECUTE-DAL-CMD SET-RECEIVE-MODE SHOW-CONNECTION-ATTRIBUTES SHOW-OUTSTANDING-MSG	These statements are permitted only between CONNECT-CONFIGURATION and DISCONNECT.

Table 12: Statement selection rules

Setting up a connection to the UDS/SQL configuration (CONNECT-CONFIGURATION)

This statement sets up a connection between UDSADM and a UDS configuration via DCAM.
If there is already a connection to the communication partner, this statement is rejected and a message output.

CONNECT**-C**ONFIGURATION
CONFIGURATION**-N**AME = <name> **,PAS**SWORD = ***NONE** / <x-string> / <c-string> **,HOST** = ***LOC**AL / <name>

CONFIGURATION-NAME = <name>
Name of the DCAM application to which UDSADM is to be connected. The name is formed from the first eight characters of the UDS/SQL configuration name.

Example

```
UDS/SQL configuration name          CONFIGURATION-NAME
UDSADMTESTDB                        UDSADMTE
UDSADM                              UDSADM
UDSA                                UDSA
```

PASSWORD = *NONE
If no administration password has been defined for the UDS/SQL configuration, you do not have to specify anything for PASSWORD.

PASSWORD = <x-string> / <c-string>
Administration password for administration via DCAM. This password may be up to four bytes long and has the following format:

X'*nnnnnnnn*'

one to eight hexadecimal digits

C'*xxxx*' one to four alphanumeric characters

If an administration password has been defined for the UDS/SQL configuration, you must specify it here.

HOST = *LOCAL
Name of the local host. If the UDS/SQL configuration to be administered runs on the local host, you do not have to specify anything for HOST.

HOST = <name>
Name of the host on which the UDS/SQL configuration to be administered runs. This name may be up to eight characters long.

Clearing a connection to the UDS/SQL configuration (DISCONNECT-CONFIGURATION)

This statement clears a connection between UDSADM and a UDS/SQL configuration.If there is no connection with a communication partner, this statement is rejected and a UDSADM message output.
If there is a connection with the communication partner, it is cleared and an acknowledgment is output.

DISCONNECT**-C**ONFIGURATION

This statement has no operands.

Terminating UDSADM (END)

This statement terminates UDSADM.
Any existing connection with the communication partner is cleared and an acknowledgment is output.

END

This statement has no operands.

Executing DAL commands (EXECUTE-DAL-CMD)

This statement sends the UDS/SQL or UDS-D DAL command specified for *dal-cmd* to the UDS/SQL configuration. UDSADM waits until UDS/SQL acknowledges receipt of the command and then issues a corresponding message.

In the case of DAL commands that require a response, UDSADM waits up to the maximum wait time set.

If the DAL response arrives within this time, it is received in its entirety and output. UDSADM then branches back to its basic state and can process further UDSADM input.

While UDSADM is waiting for the DAL response, all messages arriving from the UDS/SQL configuration are received and output.
If no DAL response arrives within the wait time, a UDSADM message is output. UDSADM branches back to its basic state and can process further UDSADM input.

The outstanding DAL response can then be received at a later time using one of the UDSADM receive functions.

EXECUTE**-D**AL**-C**MD
CMD = <dal-cmd>

CMD = <dal-cmd>

dal-cmd can be up to 64 characters long. UDSADM does not check the syntax of the DAL commands.

- If the wait time set is too short (< 5 seconds), the DCAM message buffer between the UDS/SQL configuration and UDSADM may not be large enough in the short term. Once the wait time has expired, UDSADM reports that there are no more messages. The UDSADM statement SET-RECEIVE-MODE can be used to receive the rest of the DAL result. If more than 30 seconds have passed, DAL messages may have been lost. In this case, you should repeat the DAL command with a longer wait time.
- If UDSADM is called in a procedure, you must ensure that the DAL commands for UDS-D cannot be mistaken for symbolic procedure names (see the commands manuals of "BS2000 OSD/BC", ESCAPE-CHARACTER parameter in the BEGIN-PROCEDURE command). You can do this by specifying the character "&" twice in the UDS-D DAL command.
- You should note that UDSADM interprets a final hyphen as a continuation character. This can lead to difficulties if entered DAL commands terminate with a hyphen that forms part of the DAL command (see "DAL syntax").

Help function for DAL commands (HELP-DAL-CMD)

You can use this statement to find out the meaning of individual DAL commands. The syntax of the DAL commands is described with examples.

When you enter this statement, a list of all the DAL commands is output. You can request the meaning of individual DAL commands by entering question marks.

You cannot execute DAL commands using this statement.

HELP-**D**AL-**C**MD
?

The question mark has to be specified.

Setting the message output format (MODIFY-MSG-FORMAT)

This statement allows you to set the message output format. UDSADM outputs UDS/SQL messages, UDS-D messages and DAL responses to SYSOUT in the output format set here.

MODIFY-**M**SG-**F**ORMAT
HEADER = **NO** / **YES**

HEADER = NO
The message is output to SYSOUT **without** a message header.

HEADER = YES
The message is output to SYSOUT **with** a message header; see the "Messages" manual, formatted message output.

Setting the wait time for receiving messages (MODIFY-MSG-WAIT-TIME)

This statement allows you to set the time, in seconds, that UDSADM is to wait for the arrival of a message (DAL response, UDS/SQL or UDS-D message).
If the message arrives within this wait time, it is received in its entirety and output.
If the message does not arrive within the wait time, UDSADM branches back to its basic state and can process further UDSADM input.
The outstanding message can then be received at a later time using one of the UDSADM receive functions.

MODIFY**-M**SG**-W**AIT**-T**IME
TIME = **STD** / *<integer 1..7200 seconds>*

TIME = STD

The wait time is set to the default value. The default value is 12 seconds in interactive mode or 60 seconds in batch mode.

TIME = <integer 1..7200 *seconds* **>**
The value of the wait time must be within the specified limits. Values of between 1 and 7200 seconds may be specified.

> ¡ If the setting made for the TIME parameter is too low, messages may be lost (see "Note").

Activating a permanent receive task (SET-RECEIVE-MODE)

This statement activates a receive task in UDSADM. UDSADM is now in receive mode and receives and outputs all incoming messages (DAL responses, UDS/SQL and UDS-D messages).
When UDSADM is in this state, no input is accepted. The receive task can be interrupted with the [K2] key.
UDSADM then branches back to its basic state and can process further UDSADM input.

SET**-R**ECEIVE**-M**ODE

This statement has no operands.

> ¡ UDSADM should normally be set to receive mode so that no messages are lost. If no UDSADM receive function has been used for a long time, it may be that messages are being lost during transport via DCAM.

Displaying connection information (SHOW-CONNECTION-ATTRIBUTES)

This statement is used to output information on the connection.

SHOW-**C**ONNECTION-**A**TTRIBUTES

This statement has no operands.

The connection information has the following format:

UDSADM *own-appl* IN HOST *own-name* CONNECTED TO:

CONFNAME	HOSTNAME	SECLEVEL	CON-DATE	CON-TIME
confname	*host*	*n*	*yyyy-mm-dd*	*hh:mm:ss*

The connection information should be interpreted as follows:

own-appl

DCAM application name of UDSADM in UAD@*tsn* format.
UAD@ is a fixed string.
tsn is the task sequence number of the UDSADM task.

own-name

Name of the host on which UDSADM is running.

confname

UDS/SQL configuration name; used as the DCAM application name.

host BCAM name of the host on which the UDS/SQL configuration to be administered
runs.

SECLEVEL

Security level of the UDS/SQL configuration to be administered.
This column is now only output for reasons of compatibility.

n = NO
normal security level

CON-DATE

Date of connection setup to the UDS/SQL configuration.

CON-TIME

Time of connection setup to the UDS/SQL configuration.

Displaying outstanding messages (SHOW-OUTSTANDING-MSG)

This statement is used to display outstanding messages (DAL responses, UDS/SQL and UDS-D messages).
If there are any outstanding messages when the statement is issued, they are output. Once all the messages have been output, UDSADM branches back to its basic state and can process further UDSADM input.
If there are no outstanding messages, a UDSADM message is issued. UDSADM branches back to its basic state and can process further UDSADM input.

SHOW-**O**UTSTANDING-**M**SG

This statement has no operands.

4.2 Administration of UDS/SQL via DCAM

If the DBH load parameter PP ADM=REMOTE is specified on starting the DBH, UDS/SQL opens a DCAM application. This enables the database administrator to set up a connection to UDS/SQL and enter database administrator commands from any data display terminal.

During the life of this connection the database administrator receives all UDS/SQL messages at this terminal.

Setting up the connection to UDS/SQL

You can set up a connection to the application *confname* on the BS2000 processor *processor* in a terminal emulation. If an administration password *password1* is needed, you can pass it in the user message.

Connection via OMNIS is possible using the following command (see the OMNIS manual "Functions and Commands"):

```
O[PNCON] pac,PT=confname,PR=processor,TYP=DCAM[,CMSG|LMSG=password2]
```

confname

UDS/SQL configuration name; used as the DCAM application name. It consists of the first eight characters of the configuration name in the command:

`/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=`*`configuration-name`*

processor

BCAM name of the host.

pac

Partner address code of the UDS/SQL configuration

MSG=*password1* or

CMSG=*password2* or

LMSG=*password2*

Password for administration via DCAM application. Must be specified only if a password has been defined by means of PP ADMPASS.

password1

A keyword with a length of one to four bytes

C'*xxxx*':
One through four alphanumeric characters

X'*nnnnnnnn*':
One through eight hexadecimal digits

password2

A keyword with a length of one to four bytes

xxxx
One to four characters without a data type specification and unquoted. The password must not be specified in the C'...' or X'...' format since OMNIS does not interpret these characters as format identifiers but instead passes them on unchanged in the connection message. You should also note that the LCASE parameter in OPNCON currently has no effect and that lowercase characters in MSG or LMSG are always converted to uppercase. You may need to take account of these restrictions when assigning the administration password.

Once the connection has been established, UDS/SQL considers the data display terminal in question to be an administrator terminal. A connection request is rejected if the UDSADM program or another database administrator data display terminal is already connected to the configuration. It is also not possible to use the UDSADM program or the operator console to enter UDS/SQL commands while the database administrator's data display terminal is connected via DCAM.

Once you have set up a connection to the DCAM application, you enter the command directly, i.e. without INFORM-PROGRAM or SEND-MSG:

dal-command
Pressing the [K1] key delays the sending of a UDS/SQL message by 20 seconds so that data entry is not disrupted.

Clearing the UDS/SQL connection

In order to cancel the connection to UDS/SQL, the database administrator uses the following command:

```
CLOSE ADMINSTRATION
```

The DCAM application is retained. It allows subsequent reconnection of any selected data display terminal or program.

UDS/SQL administration is performed via the local console as long as there is no existing connection via DCAM to UDS/SQL. The local console means the data display terminal on which commands are to be entered when load parameter PP ADM=LOCAL is specified, i.e.

- the operator console if the master task is running as an ENTER job.
- the data display terminal connected to the master task if the master task is running as an interactive job.

Logging administration data

If a data display terminal has set up a connection to UDS/SQL via DCAM, messages to and from UDS/SQL are also logged at the original console.

The original console is:

- the operator console when the master task is running as an ENTER job.
- the terminal connected to the task if the master task is running as an interactive job.

By means of the BS2000 command:

```
/MODIFY-JOB-OPTIONS logging=*par(listing=*yes)
```

and an appropriate assignment of the SYSLST file it is possible to store the logged data in a file.

You can use the PP UCON load parameter to define a console other than the original one as the output location for the logged data; see PP UCON in chapter "DBH load parameters".

Responses in error situations

If errors occur during UDS/SQL operation (e.g. line failure) or if termination of the DCAM application is enforced (e. g. with the network management command BCAPPL MODE=DEACTIVATE), UDS/SQL immediately tries to reopen the application. If the attempt fails, it is repeated every ten minutes. The database administrator can resume communication as soon as the application is reopened.

Messages concerning UDS/SQL DCAM administration

The DCAM administration status is logged at the local console by means of the following UDS/SQL message:

```
% UDS0347 UDS-ADMINISTRATION: ( s1) ( s2)
```

'BLANK'*s1*: Description of the event

'BLANK'*s2*: UDS/SQL diagnosis insert

The following events are reported with the appropriate additional information:

- opening of DCAM application with application name
- connection setup with terminal name
- connection cleardown with terminal name
- connection abortion with feedback information
- DCS abortion with feedback information
- SVC error in the administration routine with name of macro and error status or feedback information.

More detailed information can be found in the manuals for “DCAM (BS2000)”.

4.3 Administration of UDS/SQL via INFORM-PROGRAM

If you want to control the UDS/SQL session via INFORM-PROGRAM, you have two different entry options (see below).

Option 1: Entry from the operator console

If the DBH was started via an ENTER job, you can enter DAL commands from the operator console with the INFORM-PROGRAM command:

```
/INFORM-PROGRAM MSG='dal-command'["comment"], JOB-ID=*TSN(tsn),
```

where:

tsn Task sequence number of the master task

You must enter *dal-command* in uppercase letters. Depending on the operating system version, you may have to specify the character "&" twice. You must also specify single quotes twice.

Option 2: Entry from the database administrator's terminal

If the independent DBH was started as an interactive job from the database administrator's terminal, the entire dialog with the DBH is conducted via this terminal. During the session, the master task must perform special functions in order to guarantee that everything runs smoothly.

Care should therefore be taken to ensure that the master task is interrupted only when required and only for a brief period to enter DAL commands if it was loaded as an interactive job. You should also remember to enter RESUME-PROG after issuing BS2000 commands. Command input is accomplished by pressing the EM,DUE (Transmit) keys or the K2 key to interrupt the master task, and then passing the required DAL command to the master task by means of a INFORM-PROGRAM command:

```
/INFORM-PROGRAM MSG='dal-command'["comments"] [, JOB-ID=*OWN]
```

> The master task should not be interrupted for an extended period, since this would interrupt communication between the UDS/SQL tasks.

You must enter *dal-command* in uppercase letters. Depending on the operating system version, you may have to specify the character "&" twice. You must also specify single quotes twice.

4.4 The Database Administrator Language DAL

In a session with the independent DBH, the commands of the Database Administrator Language DAL may be used to give instructions to the independent DBH and exercise control over execution of the session.

DAL syntax

The names of DAL commands and their operands may be abbreviated in accordance with the following rules:

- Specification of the first three characters of a name is sufficient, except in the case of %BIB, %DML and %TERM.
- If more than three characters are specified, you must also specify the correct syntax for the additional characters.
- If more than three characters are specified then the DAL command entry can be abbreviated so that it ends with a hyphen contained in the DAL command. If input is performed via UDSADM, you should note that UDSADM interprets a terminating hyphen as a continuation character.
- Names which are shorter than three characters cannot be abbreviated.

For the **UDS-D**-specific DAL commands, the first three characters following "&" are sufficient.

The overall length of DAL commands must not exceed 64 characters. This may lead to problems in the case of DAL commands requiring the database name and the realm name to be specified.
Such problems can be avoided by keeping realm names unique throughout the configuration (in which case specifying the database name is superfluous) or by keeping database names and realm names to a reasonable length (see the reference section of the manual "Design and Definition").

UDS/SQL helps correct potential errors arising from DAL commands which are too long, with the UDS0220 message. On receipt of a DAL command, UDS/SQL issues the following message:

```
% UDS0220 UDS RECEIVED COMMAND
```

This message contains the DAL command accepted by UDS/SQL as insert (&01). What was actually received by UDS/SQL is apparent from this insert. In particular, it identifies whether any parts of the command which you entered were truncated.

The end of command processing for error-free DAL commands is indicated by the following message:

```
% UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND
```

or, for UDS-D DAL commands, by the following message:

```
% UDS0832 UDS-D: COMMAND EXECUTED
```

Errors in DAL commands cause the following message to be issued:

```
% UDS0209 UDS USER ERROR: COMMAND REJECTED
```

The message UDS0209 may appear more than once with differing levels of detail concerning the cause.

Certain DAL commands require *transaction-id* to be specified:

transaction-id:

> max. eight-digit decimal number assigned by the DBH to identify a transaction; it can be determined by means of the DISPLAY command.

Command execution

With most of the DAL commands the required function is executed or at least initiated immediately. With the following commands the required function is merely noted:

ACT INCR
ACT DBTT-INCR
ADD DB
ADD RN
DEACT INCR
DEACT DBTT-INCR
DROP DB
DROP RN
NEW PUBSETS
NEW RLOG
CHECKPOINT
ACCESS DB (when in combination with ADD DB)

These pending requests are not executed until a PERFORM command is issued. This enables the user to collect requests in the course of a session. When PERFORM is issued, the DBH can execute all requests in a single operation and only needs to set internal locks once.

The requests may be entered in any order, but the DBH processes them in a defined sequence. The sequence varies according to whether

- the requests refer to databases **already attached** to the current configuration at the time when the PERFORM is issued
- the requests refer to databases **not yet attached** to the current configuration at the time when the PERFORM is issued.

For example, in the case of databases which are already attached, ADD RN is executed before DROP DB. This means that any failed realms which have been repaired are made available first, and only then is the entire database detached.

The order in which such requests are executed is shown in the table below.

The request refers to databases that are already attached	The request refers to databases that are not yet attached
ACT INCR DEACT INCR ACT DBTT-INCR DEACT DBTT-INCR ADD RN= DROP RN= CHECKPOINT [DB=] DROP DB=	ADD DB= CHECKPOINT DB= ACCESS DB=

Table 13: Order of processing requests after PERFORM

The independent DBH recognizes the following DAL commands:

DAL commands for UDS-D all start with the special character &. They are evaluated in UDS-D mode only. Here they have been incorporated in the alphabetic listing of the other DAL commands.

DAL command	Meaning
ABORT {*transaction-id*[,OPTION=PTC] | ALL}	Rolls back the specified open transactions OPTION=PTC: No allowance made for interconfiguration consistency or UDS/SQL openUTM consistency
ACCESS {LOCK | RETRIEVAL | UPDATE}, {DB=*dbname* | RN=*realm-name*[,DB=*dbname*]}	Handles access locks at database and realm level
ACT {DBTT-INCR,DB=*dbname*[, RECR=*recordref*] [, EXT=*extnr*][, SCAN={YES | NO} | INCR,DB=*dbname*[, RR=*realmref*] [, EXT=*(nr-pages,min-pages)*]}	Activates online extensibility of DBTTs or realms
ADD {DB=[$*userid*.]*dbname*[.*copy-name*] [,OPTION=[SHARED-RETRIEVAL] [,OWN-BUFFER-SIZE=*n*][,ID=*bufferid*] | RN=*realm-name*[,DB=*dbname*] PW=*password* ADM=*admpassword* }	Adds databases, realms and passwords
&ADD DISTRIBUTION, {NODE=*processor-name*, CONF=*confname*[,DB=*dbname*] | DB=*dbname*,SS=*subschema-name* | FILE=*file-name* }	Adds new entries to the distribution table
%BIB	Displays the number of messages to the UDS /SQL DBH forwarded by UDSCON. Counts not only the DML statements coming from the COBOL DML, CALL DML, SQL or KDBS, but also the messages from the runtime systems and from UDSCON.
&CHANGE=DISTRIBUTION, NODE=*processor-name*, CONF=*confname*	Assigns a configuration to a different host
CHECKPOINT[DB=*dbname*[,OPTION=EVEN-WITHOUT-ALOG]]	Writes checkpoints in AFIM logging for individual databases of the configuration or for the entire configuration

CLOSE {RUN-UNITS | CALLS | ADMINISTRATION}	Terminates the current session normally or terminates DCAM administration
&CLOSE DISTRIBUTION	Terminates UDS-D operation
COMMIT *transaction-id*	Terminates transaction in PTC state and commits updates (FINISH) regardless of inter-configuration consistency or UDS/SQL-openUTM consistency
CONTINUE	Scrolls output of DISPLAY SQL DAL command
DEACT {DBTT-INCR,DB=*dbname*[, RECR=*recordref*]| INCR,DB=*dbname*[, RR=*realmref*] }	Deactivates online extensibility of DBTTs or realms
DISPLAY {DB[,RUNID=*transaction-id*] | USERS[,DB=*dbname*] | SUBSCH[,DB=*dbname*] [,LINES={*n* | ALL}] | MAINREF[,STATE=BLOCK] | *transaction-id* | REALMS[,DB=*dbname*][,RN=*realm-name*] [,LINES={*n* | ALL}] | PP | FPA [,DB=*dbname*][,RN=*realm-name*] [,LINES={*n* | ALL}] | INCR [,DB=*dbname*][,RN=*realm-name*] [,LINES={*n* | ALL}] | DBTT-INCR [,DB=*dbname* [, RECR=*recordref*]] [,LINES={*n* | ALL}] | PUBSETS | ALOG [,DB=*dbname*] }	Lists databases, transactions, subschemas, mainrefs, available free space or program parameters of the configuration or displays information on the online extensibility of realms or DBTTs or displays UDS/SQL pubset declaration
&DISPLAY DISTRIBUTION [,NODE=*processor-name*] [,CONF=*confname*] [,DB=*dbname*] [,SS=*subschema-name*]	Displays the distribution table
DISPLAY SQL{,VG=*vg-nr* | ,OPTION=ALL[,VG=*vg-nr*] | ,OPTION=IDLE[,TIME=*z*]}	Displays detailed information on one or more SQL conversations

%DML	Displays the number of messages to the UDS /SQL DBH forwarded by UDSCON. Counts not only the DML statements coming from the COBOL DML, CALL DML, SQL or KDBS, but also the messages from the runtime systems and from UDSCON.
DROP {DB=*dbname* | RN=*realm-name*[,DB=*dbname*] | PW=*password* | ADM=*admpassword*}	Detaches databases and realms and excludes passwords
&DROP DISTRIBUTION, {ALL | NODE=*processor-name*, CONF=*confname*[,DB=*dbname*][,ALL] | DB=*dbname*,SS=*subschema-name* }	Deletes entries from the distribution table
DUMP [{ALL | STD | *transaction-id*}]	Generates a DBH memory dump
EXTEND DBTT,DB=*dbname,* RECR=*recordref* [,EXT=*extnmbr*]	Executes online DBTT extension
EXTEND REALM,DB=*dbname*,RR=*realmref*, EXT=*nr-pages*	Executes online realm extension
FORGET SQL,VG=*con-no*	Releases the resources belonging to the SQL conversation numbered *con-no*
GO {*transaction-id* | ALL | OLD}	Continues execution of the specified transaction (s)
&LOCK DISTRIBUTION, {NODE=*processor-name* | CONF=*confname* | DB=*dbname* | SS=*subschema-name* }	Locks entries in the distribution table
MODIFY {ALOG | ALOG-RES}, DB=*dbname*[,VALUE={:*catid*: | PUBLIC}] [,SHARE={YES | NO}]	Changes the settings of DEFAULT-SUPPORT and RESERVE-SUPPORT of the ALOG files
MODIFY ALOG-SIZE, DB=*dbname*,VALUE=([*primary*][,[*secondary*]])	Changes the storage allocation of the ALOG files

MODIFY {LOG | LOG-2}, VALUE={:*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2 device-2* [,*priv-vsn-3 device-3*]]) | (vsn-1[,vsn-2[,vsn-3]])}	Changes the volume allocation for an original and duplicate RLOG file to be newly created
MODIFY LOG-SIZE,VALUE=([*primary*] [,[*secondary*]])	Changes the amount of storage in RLOG files
MODIFY PTCSYNCH, VALUE=([{WAIT | ABORT | COMMIT}] [,[{WAIT | ABORT | COMMIT}]])	Modifies the value of the DBH load parameter PTCSYNCH
MODIFY RESERVE, VALUE={NONE | :*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2 device-2* [,*priv-vsn-3 device-3*]]) | (vsn-1[,vsn-2[,vsn-3]])}	Changes the reserve volumes for the RLOG files
NEW PUBSETS	Checks and notes new UDS/SQL pubset declaration
NEW RLOG	Selects new RLOG files
PERFORM {NOCANCEL | CANCEL}	Initiates execution of the DROP, ADD, NEW or CHECKPOINT commands Default value: NOCANCEL
&PWD DISTRIBUTION,CONF=confname, {PWN=*new-password* | PWO=*old-password* | PWO=*old-password*,PWN=*new-password*}	Assigns and changes password
REACT INCR,DB=*dbname*[, RR=*realmref*]	Reactivates online realm extensibility
RESET ORDERS	Cancels pending requests
&SAVE DISTRIBUTION,FILE [:*catid*:][$*userid*.]*file-name*	Saves the distribution table
&START DISTRIBUTION	Starts UDS-D operation
STOP {*transaktionskennung* | NEW | ALL}	Stops execution of the specified transaction(s)

&SYNCHRONIZE DISTRIBUTION	Terminates secondary subtransactions (STTs) in the PTC state
%TERM	Aborts the session in the quickest way (emergency halt) and optionally generates a complete DBH dump ! CAUTION! %TERM should be used only if the master task stops accepting DAL commands due to an error.
&UNLOCK DISTRIBUTION, {NODE=*processor-name* | CONF=*confname* | DB=*dbname* | SS=*subschema-name* }	Removes locks on entries in the distribution table

Table 14: DAL commands for the independent DBH

The linked-in DBH recognizes the following DAL commands:

DAL command	Meaning
%BIB	Displays the number of base interface blocks processed in this part of the session
%DUMP	Outputs a complete dump of the DBH with edited tables to a file; the program is not aborted

Table 15: DAL commands for the linked-in DBH

All DAL commands are described in detail below.

4.4.1 Rolling back one or all transactions (ABORT)

```
ABORT {transaction-id[,OPTION=PTC] | ALL}
```

Default value:

without OPTION=PTC

transaction-id

Identifies an open transaction which is to be rolled back.

OPTION=PTC

Rolls back the specified transaction even if it is in the PTC state (only for UDS-D/openUTM).

ALL Rolls back all currently open transactions except for those in the PTC condition.

The ABORT command rolls back the specified transaction or all currently open transactions. The COBOL(/CALL) DML application program involved receives the status code "122". SQL programs receive a status code < -1000 (SQL code).

The DBH accepts an ABORT command only for transactions that can be rolled back without any loss of data consistency.

Without the OPTION=PTC suffix, ABORT is ignored if the transaction is in the PTC condition.

With the OPTION=PTC suffix, the database administrator rolls back a secondary subtransaction or openUTM transaction "manually", even if it is in the PTC condition and is thus waiting for a termination message from the corresponding primary subtransaction or openUTM respectively.

The effect of the DBH load parameter PP PTCSYNCH is canceled for this transaction.

If it is not possible to roll back a transaction, e.g. because the RLOG file is incorrect or incomplete, the ABORT command is ignored by the DBH.

> ¡ Rolling back a secondary subtransaction or a openUTM transaction in the PTC condition places the interconfiguration consistency or the UDS/SQL openUTM consistency at risk (see section “Terminating the PTC state”).

4.4.2 Handling access locks at database and realm level (ACCESS)

```
ACCESS {LOCK | RETRIEVAL | UPDATE},
    {DB=dbname | RN=realm-name[,DB=dbname]}
```

dbname

Name of a database in the DB configuration

realm-name

Name of a realm in the database

The ACCESS command is used to control access rights to the realms of a database or to the entire database. Accesses can be locked, restricted, or access locks can be released. The locks have only logical effect, physically the files are not disconnected.

Access locks

ACCESS LOCK,DB=*dbname*

locks the database *dbname*.

ACCESS LOCK,RN=*realm-name*, [,DB=*dbname*]

locks the realm *realm-name*. DB=*dbname* need only be specified if the realm name is not unique throughout the DB configuration. It then specifies the intended database.

The effect of ACCESS LOCK is that the DBH prohibits all further access to the specified realm or database.

ACCESS LOCK does not take effect until the next READY (status code 12022), i.e. transactions which are processing the realm or database concerned continue until FINISH. Updating accesses are however locked if READY UPDATE is used in an attempt to open a processing chain for a further realm which has already been locked against updates by means of ACCESS LOCK.

Update access locks

ACCESS RETRIEVAL,DB=*dbname*

locks the database dbname against updates

ACCESS RETRIEVAL,RN=*realm-name*, [,DB=*dbname*]

locks the realm *realm-name* against updates. DB=*dbname* need only be specified if the realm name is not unique throughout the DB configuration. It then specifies the intended database.

ACCESS RETRIEVAL locks the specified realm or database against updating transactions until the lock is released by means of an ACCESS UPDATE command.

Access is possible for read-only transactions.

ACCESS RETRIEVAL does not take effect in CODASYL applications until the next READY (status code 12123) or in SQL applications until the first updating database access, i.e. updating transactions which are processing the realm or database concerned continue until FINISH. Updating accesses are however locked if READY UPDATE is used in an attempt to open a processing chain for a further realm which has already been locked against updates by means of ACCESS RETRIEVAL. In SQL applications, updating SQL statements are rejected.

Releasing access locks

ACCESS UPDATE,DB=*dbname*

releases access locks for the database *dbname*.

ACCESS UPDATE,RN=*realm-name* [,DB=*dbname*]

releases access locks for the realm *realm-name*. DB=*dbname* need only be specified if the realm name is not unique throughout the DB configuration. It then specifies the intended database.

ACCESS UPDATE cancels a previous ACCESS LOCK or ACCESS RETRIEVAL command, i.e. it releases the lock previously imposed on the specified realm or database.

An ACCESS command for a database may also be issued between the ADD DB and PERFORM commands, in other words as soon as the database has been assigned for attachment.
This serves to ensure that the access lock defined in the ACCESS command will apply after the database has been attached.

This is advisable if, for example, transactions which update a database have to be temporarily suspended.

4.4.3 Activating online extensibility (ACT)

```
ACT {DBTT-INCR,DB=dbname[,RECR=recordref][,EXT=extnmbr][,SCAN={YES | NO} |
     INCR,DB=dbname[,RR=realmref][,EXT=(nr-pages,min-pages)]}
```

> You must enter precisely one space between ACT and DBTT-INCR or INCR. No additional spaces are permitted in this DAL command.

You can use the ACT command to activate online extensibility for realms and DBTTs.

Activating online DBTT extensibility for a record type (ACT DBTT-INCR)

dbname

Name of the database comprising the record type or types for which online DBTT extension is to be activated. The database and the corresponding realm must already be attached at the time the DAL command is issued.

recordref

Number of the user record type for which online DBTT extension is to be activated. If it is to be activated for all record types then this specification can be omitted.

You can display the *recordrefs* of the record types using the DAL command DISPLAY DBTT-INCR or take them from the BPSIA log.

recordref=1 is used internally by UDS/SQL and cannot be specified here.

extnmbr

Minimum number of entries by which the DBTT is to be extended.

This specification must be greater than or equal to 1 and less than or equal to 99999999.

The default value is the number of DBTT entries in one DBTT extension.

You can display the current value of EXT parameter using the DAL command DIS-PLAY DBTT-INCR or take it from the BSTATUS output.

As a rule the actual number of DBTT entries by which the database is extended is somewhat greater than the desired number. The actual number depends on the following factors:

- Page size of the database (2K, 4K or 8K)
- DBTT line length of the record type concerned

If bottlenecks occur in the DBTT extension, the number of entries by which the DBTT is extended can also be less than the desired number. In all cases the number of DBTT entries by which the DBTT was actually extended will be displayed with a corresponding message.

SCAN=YES

In the event of a STORE, the online DBTT extension is only performed if no further free entries can be found in the entire DBTT - unless, that is, this transaction knows from other sources that all the DBTT entries are occupied. SCAN=YES is the default setting and thus corresponds to the behavior in the case of online DBTT extensibility not being activated, where in the event of a STORE the DBTT is run through completely again before "DBTT full" is reported.

This setting may result in this particular transaction inducing a long response time and blocking other transactions.

SCAN=NO

The online DBTT extension is performed as soon as the end of the DBTT is reached on a search for free entries; existing DBTT are only ever scanned once if this setting is active. This setting is of value for record types in which records are never or only very rarely deleted or for which any gaps in the DBTT due to deletions can be tolerated.

The ACT DBTT-INCR command schedules a request for the activation of an online DBTT extension for one or more record types or cancels a preceding DEACT DBTT-INCR request.

The request is not executed until the PERFORM command is issued. The message UDS0722 informs you of request execution as follows:

- If an activation request has been executed for all record types then a single message is output for the database.
- If an activation request has been executed for a single record type then a record typespecific message is output.
- If activation requests have been executed for individual record types as well as for all record types then a single message is output for the database.

If multiple ACT DBTT-INCR requests are issued for one and the same record type then the specifications in the last issued request apply.

A DEACT DBTT-INCR request cancels a preceding ACT DBTT-INCR request for a record type and vice versa. This allows you to undo incorrectly issued ACT DBTT-INCR or DEACT DBTT-INCR requests before they are made effective with PERFORM.

The online extension of a record type remains activated until it is deactivated with DEACT DBTT-INCR.

If an online DBTT extension fails because of a suspended realm extension then a maximum of one message per record type is output to inform you that the online DBTT extension has failed. Any other messages relating to failed online DBTT extensions are suppressed until this suspension of the online realm extension has terminated.

If, on the activation of an online DBTT extension without a *recordref* specification i.e. activation for all the record types in a database, a realm with DBTTs is not attached, then the activation is not carried out for the corresponding record types. The message UDS0745 informs you of the realm in question.

If, on the activation of an online DBTT extension without a *recordref* specification i.e. activation for all the record types in a database, online realm extension is not activated for a realm with DBTTs or an online realm deactivation request is already present, then the activation is not carried out for the corresponding record types. The message UDS0744 informs you of the realm in question.

The ACT DBTT-INCR command is immediately rejected with message UDS0209 if

- the specified record type number does not exist.
- the specified database is not attached or is attached in SHARED-RETRIEVAL mode.
- the corresponding DBTT realm is not attached for a specified record type. The message UDS0745 informs you of the realm in question.
- online realm extension is not active for a specified record type or if there is already a request for the deactivation of the online realm extension of the DBTT realm. The message UDS0744 informs you of the realm in question.

- an activation request for online DBTT for all record types in a database does not result in activation or the cancellation of a contrary request for any record type.

Activating online extensibility for a realm (ACT INCR)

dbname

Name of the database of the realm or realms for which online extensibility is to be activated.
dbname must be active when the command is issued.
dbname must not be in SHARED-RETRIEVAL mode.

realmref

Number of the realm for which online extensibility is to be activated. The realm must be active when the command is executed. The numbers of the activated realms can be displayed with DISPLAY INCR.

The numbers of all user realms and of the DBDIR are permitted. It is not permissible to specify the number of the DBCOM.

If RR=*realmref* is not specified, the command is executed with the same values for all user realms which have just been activated and for the DBDIR of the database *dbname*.

nr-pages

Number of database pages to be appended to the realm during online realm extension.
nr-pages=64 ... 16777215
The default value is 64.

nr-pages is increased by a maximum of 64 by the DBH during online realm extension, if space for a new FPA extent has to be deducted from the extension space.

Irrespective of *nr-pages*, an online realm extension is limited to the maximum number of possible database pages in the realm.

nr-pages does not relate to the number of pages by which the realm is extended by the DMS, but to the number of database pages.

In the case of realms that only contain DBTTs and are therefore only activated for online DBTT extension, it is not necessary to specify *nr-pages* since UDS/SQL extends the realm on the basis of internal rules.

min-pages

Limit value for free database pages in realm.
min-pages=0 ... *nr-pages*
The default value is 16.

min-pages has the following effect:

min-pages=0

Realm extension is started if a free space search has been unsuccessful.

min-pages>0

If the number of currently existing free database pages falls below *min-pages*, then a free space search will result in a realm expansion.

Realm extension will be performed, even if it is not yet necessary for carrying out the current request. This enables further transactions requiring free place in the realm to be performed during online realm extension.

In the case of realms that only contain DBTTs and are therefore only activated for online DBTT extension, it is not necessary to specify *min-pages* since UDS/SQL extends the realm on the basis of internal rules.

The command ACT INCR will note a request to activate online extensibility, for the realm *realmref* in the database *dbname*. If RR=*realmref* is not specified, the request will be noted to activate online extensibility for all activated realms in the database *dbname*.

ACT INCR can also undo a previously issued DEACT INCR command, provided the commands have not already been executed with PERFORM. ACT INCR will only ever be executed once PERFORM has been specified. If you have issued several ACT INCR requests for a single realm, the data of the last request applies.

DAL requests pending in addition to ACT INCR may also contain requests which will drop the realms or databases affected. In this case, it does not matter if the DROP requests are entered before or after the ACT INCR requests. When PERFORM is issued, ACT INCR requests for a database are always processed first.

If online extensibility was suspended for the specified realm(s), ACT INCR will cancel the suspension, i.e. ACT INCR includes the function of the REACT INCR command.

The online expandability of a realm remains activated until it is deactivated by the command DEACT INCR.

If online realm extension is active then the termination of suspension may result in messages reporting the failure of DBTT extensions in this realm if a renewed suspension occurs.

> **i** If a realm is to be extended online, it is your responsibility as an administrator to allocate adequate storage space or a secondary allocation greater than 0 to the realm in sufficient time. For reasons relating to DMS, this can only be performed offline. The DBH does not check this prerequisite (see also chapter “Resource extension and reorganization during live operation”.)

The command ACT INCR is rejected immediately, if:

- an invalid value was specified for *nr-pages* or *min-pages*.
- the database *dbname* has not been activated, or has been activated in SHARED-RETRIEVAL mode.
- the specified realm number *realmref* does not exist.
- the DBCOM realm number was specified as the realm number *realmref*.
- the specified realm has not been activated.

4.4.4 Attaching databases, realms and passwords (ADD)

```
ADD {DB=[$userid.]dbname[.copyname][,OPTION=[SHARED-RETRIEVAL]
        [,OWN-BUFFER-SIZE=n][,ID=bufferid] |
     RN=realm-name[,DB=dbname]
     PW=password
     ADM=admpassword }
```

$*userid*

User identification to which the database is assigned. Only required if the database is not cataloged under the identification with which the DBH was started.

dbname

Name of the database

copy-name

Suffix of the shadow database.
If *copy-name* is specified, the corresponding shadow database is attached.

n Size of the user buffer pool for the database in Mbytes. If the parameter *bufferid* is specified, the buffer pool specified by *n* is created as a shared user buffer pool (see below):

n = 1 .. 2047

For a database in 8-Kbyte page format, at least 3 Mbytes are created.

n = 0

The size of the user buffer pool has already been defined in another database.

n not specified and *bufferid* not specified

The database is buffered in the system buffer pool corresponding to its page format. If this system buffer pool does not exist, activation of the database is rejected.

n not specified and *bufferid* specified

The size of the user buffer pool has already been defined in another database.

bufferid

Identifier of a shared user buffer pool up to 6 alphanumeric characters (no special characters)
If *bufferid* is specified, the database is assigned the shared user buffer pool identified with *bufferid.*

realm-name

Name of the realm

password

password may be up to four bytes in length and is represented as follows:
C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

d:
where *d* is a decimal number (comprising up to eight digits and a sign) to be converted to a binary value
The syntax conventions of BS2000 apply (see the commands manuals for "BS2000 OSD/BC", ADD-PASSWORD).

admpassword

admpassword may be up to four bytes in length and is represented as follows:
C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

The ADD command is used to link databases, realms and passwords into an ongoing session. However, it is also possible to use an ADD command to revoke a DROP command, provided that no PERFORM command has yet been issued to initiate execution of the DROP and ADD commands. As a rule, ADD DB and ADD RN are not executed until a PERFORM has been issued.

Attaching databases

ADD DB=[$*userid*.]*dbname*

The DBH notes a request to attach the given original database. As a result, the current DB configuration is altered. If the specified database is already attached and a DROP command requesting detachment of the database has previously been issued, the DROP command is revoked by ADD DB.

ADD DB=$*userid*.*dbname*.*copy-name*

Attaches the corresponding shadow database, and thus alters the current DB configuration.

When a detach request (DROP DB) exists for a database which has already been attached (original or copy), this detach request is only canceled by a subsequent DAL command ADD DB if the *copyname* specified in ADD DB matches the copy currently attached (in the case of a DB original this means: no *copyname* is specified). Otherwise the current copy of the database is detached with the following PERFORM, and the copy with the specified copy name is attached.

This enables

- a DB original to be replaced by a DB copy,
- a DB copy to be replaced by a DB original or
- a DB copy to be replaced by another DB copy

during PERFORM processing (i.e. in one step).

OPTION default value

With regard to the original database:
No RETRIEVAL option, i.e. the DBH opens the database in the exclusive mode; the application programs have read and write access.
With regard to the shadow database: SHARED-RETRIEVAL.

OPTION=SHARED-RETRIEVAL

The DBH has shared access to the added database. Other DBHs may also access this database via OPTION=SHARED-RETRIEVAL. Application programs of all these DBHs will then have only read access to the added database.

> ¡ To prevent the command ADD DB from exceeding the total permitted length of 64 characters, you should use the abbreviation SHA to enter the keyword SHARED-RETRIEVAL.

OWN-BUFFER-SIZE=*n*

Specifies the size of a user buffer pool for the database to be attached. If a database with an 8-Kbyte page format is involved, at least 3 Mbytes are allocated. The buffer pool is created dynamically on attaching the database in addition to the system buffer pools. If ID=*bufferid* is not specified, this buffer pool should only be used to buffer pages of the associated database. If ID=*bufferid* is specified, a shared user buffer pool is allocated to the database (see below).

If OWN-BUFFER-SIZE is not specified, the database is buffered in the system buffer pool for its page format. If this system buffer pool does not exist, the request to attach the database is rejected.

ID=*bufferid*

The buffer pool specified with OWN BUFFER-SIZE=*n* is assigned to the database as shared user buffer pool with the identifier *bufferid*. A shared user buffer pool can be assigned to several databases as an exclusive, shared buffer pool. If no buffer pool with the identifier *bufferid* is available, a new shared user buffer pool with this name will be created and assigned to the database. Otherwise, the buffer pool *bufferid* is also used to buffer the pages of the associated database, under the following conditions:

- The page size of the database matches the page size of the shared user buffer pool
- The size of the buffer pool specified with OWN-BUFFER-SIZE=*n* corresponds to the size of the available shared user buffer pool
  or
  OWN-BUFFER-SIZE has the value 0
  or
  OWN-BUFFER-SIZE=*n* is not specified.

Responses in error situations

The ADD DB command is immediately rejected if

- the number of databases specified for attachment is greater than that specified in PP MAXDB
- an identically named original database or shadow database is already attached and no request to detach (DROP) it has been issued
- a request to attach (ADD) an identically named database has already been issued.

- the size of the buffer pool specified with OWN-BUFFER-SIZE=*n* does not correspond to the size of the shared user buffer pool specified with ID=*bufferid*.
- OWN-BUFFER-SIZE=*n* has the value 0, ID=*bufferid* is specified and the *bufferid* buffer pool is not yet defined (in another database).

During processing of the request to attach a database there may be errors if

- the specified database does not exist or cannot be attached in the required usage mode.
- the specified database is inconsistent and consistency cannot be restored by means of a warm start because the database is engaged in RETRIEVAL functions or because the DBH cannot access the associated RLOG file and /or DB status file exclusively (see section "Evaluation of DBH load parameters" in chapter "DBH load parameters").
- owing to distributed processing featuring UDS-D/openUTM in the (inconsistent) database there are transactions in the "Prepared to Commit" (PTC) condition and, in the case of UDS-D, the PTCSYNCH value for the current session is set to WAIT.
- errors occur when the files of this database are attached.
- the page size of the database does not correspond to that of the shared user buffer pool specified with ID= *bufferid*.
- one of the ALOG logging specifications which were made using the DEFAULT-SUPPORT or RESERVE-SUPPORT parameter in the BMEND statement START-LOG is outside the pubset space of the current UDS /SQL pubset declaration.

If consistency cannot be restored owing to one of these errors occurring while the database is being attached, the database is immediately detached again.

If database consistency can be restored in spite of the above problems, the database is attached to the fullest possible extent to enable at least a partial realm configuration to be deployed.

Attaching realms

ADD RN=*realm-name[*,DB=*dbname*]

> The DBH notes a request to attach the realm *realm-name* or cancels a pending DROP command.
>
> DB=*dbname* need only be specified if the realm name is not unique throughout the DB configuration. It then specifies the intended database.

The ADD RN command can be used to rejoin to the current session a realm detached (DROP) for repairs or owing to device shortages.

The ADD RN request is not executed until the PERFORM command is issued.

The ADD RN command can be used to revoke an earlier DROP RN command. DROP requests which have already been started cannot be revoked.

Responses in error situations

The ADD RN command is rejected if:

- the designated realm is not present in the attached databases
- the realm name is not unique in the current DB configuration because DB=*dbname* has not been specified
- the designated realm is already attached and no request to detach it has been issued

- the designated realm cannot be attached because it is a temporary realm or part of a database for which SHARED RETRIEVAL has been specified
- errors occur when the realm is being attached
- the realm is marked defective or inconsistent.

A failsafe log of the attachment of a realm is not entered in the DBDIR until the request has been processed in full. A failsafe attachment is one that remains in effect following session aborts and DBH termination.

Attaching file passwords

ADD PW=*password*

Attaches the designated file password. This alters the current set of passwords.

Responses in error situations

The ADD PW command is rejected if the limit of 100 concurrent passwords is exceeded as a result of adding the designated password (see section “Assigning passwords to UDS/SQL files”).

Attaching administration passwords

ADD ADM=*admpassword*

Attaches the designated database administrator password. As a result the database administrator password is either specified for the first time if no password has yet been attached by means of the PP ADMPASS parameter, or changed, provided that the current database administrator password has previously been excluded with the DAL command DROP ADM.

Responses in error situations

The ADD ADM command is rejected if an database administrator password has already been defined for the DBH.

4.4.5 Adding new entries to the distribution table (&ADD DISTRIBUTION)

For UDS-D

```
&ADD DISTRIBUTION,
   {NODE=processor-name,
         CONF=confname[,DB=dbname] |
    DB=dbname,SS=subschema-name |
    FILE=file-name }
```

processor-name

Name of a host;

must be no more than eight characters in length.

confname

The first eight characters of *configuration-name*; must be unique in its first seven characters. The eighth character must not be '@'.

Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

dbname Name of a database;

may be up to 17 characters in length;

must be unique network-wide.

subschema-name

Name of a subschema;
must be no more than 30 characters in length;
must be unique network-wide in its first six characters.

Only user subschemas may be specified.
The PRIVACY-AND-IQF subschema and COMPILER subschema are not permitted in the distribution table.

:*catid*: BS2000 catalog identifier

In this case specifying :*catid*: is permitted (see section "Using pubsets in UDS/SQL").

userid User identification to which *file-name* is assigned

file-name

Name of the file containing the entries to be added to the existing distribution table. Configuration-based passwords, if any, are read in from this file.

This command adds new entries to the distribution table (see section "Structure of the distribution table").

`NODE=...,CONF=...`

These operands are used to specify a NODE/CONF entry.

The command is rejected if this configuration is already assigned to another host. In that case, the DAL command &CHANGE DISTRIBUTION should be used.

`NODE=...,CONF=...,DB=...`

These operands are used to specify a DB entry.

The DB entry is chained to the associated NODE/CONF entry if the latter already exists or the NODE /CONF entry is newly created if the specified entry does not yet exist. The two newly created entries are chained together.

`DB=...,SS=...`

These operands are used to specify an SS entry.

The SS entry is chained to the associated DB entry if the latter already exists or a new DB entry is created if the specified entry does not yet exist. The two new entries are chained together.

`FILE=...`

This operand is used to add the entries in this input file to the distribution table.

`&ADD DISTRIBUTION` only changes the local distribution table.

You cannot assign configuration-based passwords with this DAL command. For information on assigning them, see section “Configuration-based password protection” and the description of the DAL command &PWD DISTRIBUTION on "Assigning and changing a password (&PWD DISTRIBUTION)".

> ¡ Changes to the distribution table do not take effect until the next READY statement is issued.

Example

DAL command	Content of distribution table
`&ADD DIS,NODE=PROC3,CONF=CONFD`	`PROC3/CONFD`

`&ADD DIS,NODE=PROC3,CONF=CONFD,DB=DBD1`	`^`
	`|`
	`DBD1-------+`

`&ADD DIS,NODE=PROC4,CONF=CONFE,DB=DBE1`	`DBE1D-->PROC4/CONFE`

`&ADD DIS,DB=DBE1,SS=SSE11`	`^`
	`|`
	`SSE11---+`

`&ADD DIS,DB=DBF1,SS=SSF11`	`SSF11-->DBF1`

4.4.6 Displaying the number of processed base interface blocks (%BIB)

```
%BIB
```

The %BIB command causes the number of base interface blocks processed in the current section of the session to be displayed on the database administrator’s data display terminal or the operator console.

The %BIB command also applies to the linked-in DBH.

This command also displays the number of COBOL/CALL DML statements processed by a remote configuration and the number of COBOL/CALL DML statements processed by this configuration for remote application programs.

4.4.7 Assigning a configuration to a different host (&CHANGE DISTRIBUTION)

For UDS-D

```
&CHANGE DISTRIBUTION,NODE=processor-name,CONF=confname
```

processor-name

Name of a host;
must not be more than eight characters in length.

confname

The first eight characters of *configuration-name*; must be unique in its first seven characters. The eighth character must not be '@'. Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

processor-name is changed in the NODE/CONF entry in the distribution table. After the DAL command has been executed, UDS-D directs DML statements referencing the configuration *confname* to the host *processor-name*.

&CHANGE DISTRIBUTION only changes the local distribution table.

> ¡ Changes to the distribution table do not take effect until the next READY statement is issued.

4.4.8 Writing checkpoints (CHECKPOINT)

```
CHECKPOINT[ DB=dbname[,OPTION=EVEN-WITHOUT-ALOG]]
```

dbname Name of the database for which a checkpoint is to be written.

CHECKPOINT
(without DB)

Checkpoints are to be written for all databases in the configuration.

OPTION=EVEN-WITHOUT-ALOG

A checkpoint is also written on the *dbname* database if it is operated without AFIM logging. If OPTION=EVEN-WITHOUT-ALOG is not specified, the CHECKPOINT command is rejected for such databases. Specification of OPTION=EVEN-WITHOUT-ALOG has no effect for databases with AFIM logging.

The CHECKPOINT command is used to issue a request to write a checkpoint. To write checkpoints the DBH closes the ALOG file for each database and then opens a new ALOG file. The request is not executed until a PERFORM command is issued.

All RETRIEVAL transactions, including any newly initiated ones, continue without interruption. All current UPDATE transactions are completed. With COBOL DML, generally all UPDATE transactions initiated while checkpoints are being written are rejected with status code 12122. This status code can be influenced using PP ORDER-DBSTATUS.

New SQL transactions can execute as RETRIEVAL transactions. UPDATE statements are rejected (SQL CODE -810).

A CHECKPOINT command is rejected if AFIM logging is activated for the databases involved and the databases were attached with the SHARED-RETRIEVAL option.

The CHECKPOINT command can also be issued for a database which is not yet attached to the DB configuration but has been specified in a request for database attachment. This is advisable in order to ensure that checkpoints can be written even for databases which are attached by means of the DAL command ADD DB. The DBH load parameter PP STDCKPT is not applicable to dynamic attachment and detachment of databases.

> The DBH can open a new ALOG file only if the file was created previously by means of the CREATE-FILE command or if the database resides under the configuration user ID.

If the DBH cannot make a new ALOG file ready for use, it internally locks the affected database against being accessed for updating (UPDATE), i.e. it permits access only for reading.
This lock **cannot** be removed by the database administrator by means of the DAL command ACCESS UPDATE.

However, the database administrator can generate (a new version of) the missing ALOG file using the CREATE-FILE command and enter another CHECKPOINT command and another PERFORM for the database, thereby releasing the internal UPDATE lock. See also "Switching the ALOG file" and "ALOG file overflow"on page 127.

If a CHECKPOINT request is executed without specifying DB=*dbname* and databases are attached to the configuration for which no AFIM logging is activated, a message will be issued for each of these databases indicating that no checkpoint was written.

Responses in error situations

The CHECKPOINT command is rejected if

- the database is run with OPTION=SHARED-RETRIEVAL.
- no database with the given name exists and no request to attach a database with that name is pending.
- a database with the given name does exist but is run without AFIM logging, and OPTION=EVEN-WITHOUT-ALOG is also not specified.

4.4.9 Terminating the session or administration (CLOSE)

```
CLOSE {RUN-UNITS | CALLS | ADMINISTRATION}
```

The CLOSE command terminates a session or the DCAM administration normally (see section "Clearing a connection to the UDS/SQL configuration (DISCONNECT-CONFIGURATION)" in chapter "UDSADM statements").

RUN-UNITS

Initiates the normal termination procedure:

All active transactions can continue working normally, but the DBH will not allow any new transactions (status code 151).

When all transactions have finished, the master task terminates all server tasks. The master task then closes the session with the following message:

```
%UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE
%UDS0213 UDS NORMAL SYSTEM TERMINATION
```

Transactions in the STOP condition are activated automatically (see the STOP command in chapter "Stopping the execution of transactions (STOP)").

The number of DML statements per database and the number of input and output operations per database are output to SYSOUT at the following opportunities:

- When the database is dropped (DROP DB)
- As soon as the DBH has been terminated

CALLS

Effects fast termination of a session:

The DBH rolls back all open transactions (status code 122, SQL code -1700). Any attempt to start a transaction is rejected with status code 151 or 122 (SQL code -1820 or -1700).

Note on UDS-D

A rollback is not performed if the transaction is in the PTC condition. For CLOSE CALLS to take effect, transactions in the PTC condition must be terminated individually (see section "Terminating the PTC state").

ADMINISTRATION

The current administration via DCAM is terminated. Administration via DCAM can be started again if the need arises.

4.4.10 Terminating UDS-D operation (&CLOSE DISTRIBUTION)

For UDS-D

```
&CLOSE DISTRIBUTION
```

UDS/SQL is requested to terminate UDS-D operation, i.e. to terminate the UDS-D task UDSCT. The UDS-D task is not terminated until all secondary subtransactions in the local configuration have been terminated or have been in the PTC state for a period of time determined by the value of the DBH load parameter PP CHCKTIME (see section “Terminating UDS-D mode”).

When &CLOSE DISTRIBUTION is specified, READY statements from remote application programs are rejected and no distributed transactions from local users are accepted.

The application program receives the status code 141.

Once the UDS-D task UDSCT has been terminated, UDS-D mode ends in the current configuration, i.e. no further remote messages are received, and no further messages are sent to remote configurations.

4.4.11 Terminating a transaction in the PTC state (COMMIT)

For UDS-D/openUTM

```
COMMIT transaction-id
```

transaction-id

designates a secondary subtransaction or openUTM transaction in the PTC state that is to be terminated.

COMMIT is used to terminate a secondary subtransaction or openUTM transaction in the PTC state and to commit the changes to the database (FINISH).
If the specified transaction is not in the PTC state, UDS/SQL rejects the command.

The effect of the DBH load parameter PP PTCSYNCH is canceled for this transaction.

> ¡ Terminating a secondary subtransaction or openUTM transaction in the PTC condition puts interconfiguration consistency and UDS/SQL openUTM consistency respectively at risk (see section “Terminating the PTC state”).

4.4.12 Continuing DISPLAY SQL output (CONTINUE)

```
CONTINUE
```

This function enables you to "scroll" through the output of the DAL command DISPLAY SQL. When there are too many conversations for all of them to be displayed on one screen, CONTINUE can be used to continue the output.

It must be borne in mind that it is always the current state that is displayed. This means that the values may change between the time the DISPLAY SQL command is entered and the time the CONTINUE command is entered.

For example, if a new conversation has been opened since the DISPLAY SQL command was entered, when a CONTINUE command is entered an additional conversation will be shown.

CONTINUE always refers to the last "scrollable" DISPLAY SQL command entered, even if other DAL commands have been entered since then.

Output formats

The output generated by the CONTINUE command is described under the various DISPLAY SQL commands.

4.4.13 Deactivating online extensibility (DEACT)

```
DEACT {DBTT-INCR,DB=dbname[,RECR=recordref]|
       INCR,DB=dbname[,RR=realmref] }
```

> You must enter precisely one space between DEACT and DBTT-INCR or INCR. No additional spaces are permitted in this DAL command.

You use the DEACT command to deactivate online extensibility for DBTTs and realms.

Deactivating online DBTT extensibility for a record type (DEACT DBTT-INCR)

dbname Name of the database comprising the record type or types for which online DBTT extension is to be deactivated. The database and the corresponding realm must already be attached at the time the DAL command is issued.

recordref

Number of the user record type for which online DBTT extension is to be deactivated. If it is to be deactivated for all record types then this specification can be omitted.

You can display the *recordrefs* of the record types using the DAL command DISPLAY DBTT-INCR or take them from the BPSIA log.

recordref=1 is used internally by UDS/SQL and cannot be specified here.

The DEACT DBTT-INCR command is used either to note a request for the deactivation of online DBTT extension or to cancel a preceding ACT DBTT-INCR request.

The request is not executed until the PERFORM command has been issued. The message UDS0722 informs you about command execution as follows

- If a deactivation request has been executed for all record types then a single message is output for the database.
- If deactivation request has been executed for a single record type then a record typespecific message is output.
- If deactivation requests have been executed for individual record types as well as for all record types then a single message is output for the database.

If, on the deactivation of an online DBTT extension for all the record types in a database, a realm with DBTTs is not attached, then the deactivation is not carried out for the corresponding record types. The message UDS0744 informs you of the realm in question.

With DEACT DBTT-INCR, extent size, specified previously in ACT DBTT-INCR EXT=*extnmbr* is reset.

The command DEACT DBTT-INCR is immediately rejected with the message UDS0209 if

- the specified record type number does not exist.
- the specified database is not attached or is attached in SHARED-RETRIEVAL mode.
- the corresponding DBTT realm is not attached for a specified record type. The message UDS0745 informs you of the realm in question.
- a deactivation request for online DBTT extension for all record types in a database does not result in deactivation or the cancellation of a contrary request for any record type.

Deactivating online extensibility for a realm (DEACT INCR)

dbname

Name of the database of the realm or realms for which online extensibility is to be deactivated. The database must be active when the command is issued. The database must not be in SHARED-RETRIEVAL mode.

realmref

Number of the realm for which online extensibility is to be deactivated. The realm must be active when the command is executed. The numbers of the active realms can be displayed with DISPLAY INCR.

The numbers of all user realms and of the DBDIR are permitted. It is not permissible to specify the number of the DBCOM.

The command DEACT INCR can note a request to deactivate online extensibility, for the realm *realmref* of the database *dbname*. If RR=*realmref* is not specified, the command is executed for all user realms which have just been activated and for the DBDIR of the database *dbname*.

DEACT INCR can also cancel a previously entered ACT INCR command, provided that the commands have not already been executed with PERFORM. DEACT INCR is only ever executed once PERFORM has been specified.

DAL requests pending in addition to DEACT INCR may also contain requests to drop the relevant realms or databases. It does not matter whether these DROP requests are entered before or after the DEACT INCR. When PERFORM is issued, the DEACT INCR requests for a database are always processed first. For information on executing pending DAL commands, see also "Command execution" in chapter "The Database Administrator Language DAL".

If the specified realm(s) cannot be extended online, DEACT INCR is ignored.

Any outstanding EXTEND-DBTT requests that have not yet been executed since no record has been stored since the request was noted are canceled without any notification to the user when DEACT INCR is run.

The command DEACT INCR is immediately rejected, if:

- the database *dbname* has not been activated or has been activated in SHARED-RETRIEVAL mode.
- the specified realm number *realmref* does not exist.
- the DBCOM realm number was specified as the realm number *realmref*.
- the specified realm has not been activated.
- at least one online DBTT extension is activated for the realm or an online DBTT extension request has already been noted for the realm. Message UDS0738 informs you of the realm in question.

4.4.14 Listing information on the DB configuration (DISPLAY)

```
DISPLAY {DB[,RUNID=transaction-id] |
         USERS[,DB=dbname] |
         SUBSCH[,DB=dbname][,LINES={n | ALL}] |
         MAINREF[,STATE=BLOCK] |
         transaction-id |
         REALMS[,DB=dbname][,RN=realm-name][,LINES={n | ALL}] |
         PP |
         FPA [,DB=dbname][,RN=realm-name][,LINES={n | ALL}] |
         INCR [,DB=dbname][,RN=realm-name][,LINES={n | ALL}] |
         DBTT-INCR [,DB=dbname[, RECR=recordref]][,LINES={n | ALL}] |

         PUBSETS |
         ALOG [,DB=dbname] }
```

transaction-id

Identifier for a current transaction

dbname

Name of a database in the DB configuration (or name of a shadow database)

In the case of DISPLAY DBTT-INCR, name of the database containing the record type or record types for which the user schema information is to be output. If the DB is not specified then all the record types of all the currently attached databases are output.

realmname

Name of a realm

recordref

Number of the user record type for which information is to be output.
If the RECR parameter is not specified then all the record types for the user schema in the specified database are output.

The DISPLAY DBTT-INCR command with no *recordref* specification is particularly useful if you want to assign record numbers to record type names.

recordref=1 is used internally by UDS/SQL and cannot be specified here.

n Maximum number of output lines for this DISPLAY command (default: 100). You can specify values between 5 and 999 for *n*.
Limiting the number of output lines helps you prevent the disruption of DBH operation due to the unintentional generation of a very large number of output lines.

ALL All the output lines are generated.

The DISPLAY command produces a listing on the database administrator's data display terminal or operator console in accordance with the command specification, i.e.:

DB Display all databases in the current DB configuration

DB,RUNID=*transaction-id*

Display all databases referenced by the transaction *transaction-id*

USERS

Display all transactions currently running in the DB configuration

USERS,DB=*dbname*

Display all transactions which reference the database *dbname*

SUBSCH

Display all subschemas of the DB configuration

SUBSCH,DB=*dbname*

Display all subschemas of database *dbname*

MAINREF

Display information on the status of the transaction channels (mainrefs)

MAINREF,STATE=BLOCK

Display information on status of blocked mainrefs

transaction-id

Display information on status of transaction *transaction-id*

REALMS

Display information on all permanent realms of the DB configuration

REALMS,DB=*dbname*

Display information on all permanent realms of database *dbname*

REALMS,RN=*realm-name*
[,DB=*dbname*]

Display information on the permanent realm *realm-name* [of database *dbname*]

PP Display all load parameters under which the DBH is currently running

FPA Display information on the free pages in all realms of the DB configuration

FPA,DB=*dbname*

Display information on the free pages in all realms of database *dbname*

FPA,RN=*realm-name*

[,DB= *dbname*]
Display information on the free pages in the realm *realm-name*
[of database *dbname*]

INCR Information on the online extensibility of all activated user realms and all DBDIRS in the DB configuration

INCR, DB=*dbname*

Information on the online extensibility of all activated user realms and the DBDIR of the database *dbname*

INCR,RN=*realmname*
[,DB=*dbname*]

Information on the online extensibility of the realm *realm-name* of the database *dbname*. If *realm-name* is unique in the database configuration, the specification DB=*dbname* need not be made.

DBTT-INCR

Information on the online DBTT extensibility of the record types in the user schemata of the DB configuration

DBTT-INCR, DB=*dbname*

Information on the online DBTT extensibility of the record types in the user schema of the database *dbname*

DBTT-INCR,DB=*dbname*,RECR=*recordref*

Information on the online DBTT extensibility of the record type with record type number *recordref* in the database *dbname*

PUBSETS

Settings of the UDS/SQL pubset declaration

ALOG

Current ALOG settings on all ALOG files of the DB configuration.

ALOG DB=*dbname*

Current ALOG settings of database *dbname*.

LINES

Number of output lines for this DISPLAY command.

LINES=*n*

Maximum number of output lines for this DISPLAY command.

If the maximum number of output lines for a DISPLAY command is not sufficient to output the information for all the objects in question then a corresponding message is output:

```
OUTPUT OF nnnn FURTHER LINES SUPPRESSED
```

There is no command allowing you to resume this interrupted DISPLAY command. To obtain the information for all the associated objects, you must repeat the command, for example with LINES=ALL.

In all cases, the following message is then output:

```
UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND
```

LINES=ALL

All output lines for this DISPLAY command.

The following sections describe the output of the various DISPLAY commands.

DISPLAY DB - Show all databases and access modes

Input format: DISPLAY DB

```
Comment line:  USERID:   DATABASE-NAME:  DB-OPTION:  BUFFER: BUF-ID: PAGE:
Information :  userid    dbname          db-option   bbbb    buffid  pgKB
line(s)
End line:       UDS/SQL Vn.n   nnn  DATABASE(S) IN confname
```

Explanations:

userid User ID in which the database concerned is located.

dbname

Name of the database

db-option

EXC-UPD
EXCLUSIVE-UPDATE:
The DBH accesses database *dbname* exclusively. The application programs can access the database in update mode.

SHA-RTR
SHARED-RETRIEVAL:
All DBHs can have shared access to database *dbname*, i.e. this database can also be attached to other DBHs (but only in SHA-RTR mode). The application programs of all DBHs have read-only access to the database.

bbbb Size of the user buffer pool for the database in Mbytes without leading zeros. If no user buffer pool exists, blanks are output.

buffid Name of the shared user buffer pool
If a shared user buffer pool has been assigned to the database.

*EXCL
indicates that an exclusive user buffer pool has been assigned to the database.

*SYSTEM
indicates that the system buffer pool has been assigned to the database.

pg Page format of realm files

V*n.n* UDS/SQL version number

nnn Number of databases in the DB configuration

Example

//EXECUTE-DAL-CMD CMD=DISPLAY DB

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY DB (OPOX073,09:28:51/4TMW)
4TMW: USERID:   DATABASE-NAME:      DB-OPTION: BUF-SIZE: BUF-ID: PAGE:
4TMW: ------------------------------------------------------------------
4TMW: $XXXXXXXX.SHIPPING            EXC-UPD            *SYSTEM  4KB
4TMW: $XXXXXXXX.CUSTOMER            EXC-UPD       1    *EXCL    4KB
4TMW:
4TMW: UDS/SQL V2.9  2 DATABASES IN CONFEXMP
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:51/4TMW)
```

DISPLAY DB,RUNID=*transaction-id* - Show all databases referenced by the transaction identifier

Input format: DISPLAY DB,RUNID=*transaction-id*

```
Comment line:   USERID:   DATABASE-NAME:          OPEN-MODE:


Information
line(s):        userid    dbname                  open-mode

End line:       UDS/SQL Vn.n nnn DATABASE(S) USED FROM transaction-id
```

Explanations:

transaction-id

Internal transaction identifier assigned by UDS/SQL.

userid User ID in which the database concerned is located.

dbname

Name of the database

open-mode

Usage mode specified by the transaction *transaction-id* when opening the realm(s) of the database:

UPD
UPDATE

RTR
RETRIEVAL

V*n.n* UDS/SQL version number

nnn Number of databases referenced by transaction *transaction-id*

Example

//EXECUTE-DAL-CMD CMD=DISPLAY DB,RUNID=1

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY DB,RUNID=1 (OPOX073,09:28:51/4TMW)
4TMW: USERID:   DATABASE-NAME:      OPEN-MODE:
4TMW: ----------------------------------------
4TMW: $XXXXXXXX.SHIPPING             RTR
4TMW:
4TMW: UDS/SQL V2.9   1 DATABASE  USED FROM       1
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:51/4TMW)
```

DISPLAY USERS - Show all transactions currently running

Input format: DISPLAY USERS

```
Comment line:      PROGRAM  TSN/TERM  RUNUNIT-ID  STATE  PTC  FC  MR-NR  DLY


Information
line(s):           progname tsn/term  ta-id       state  ptc  fc  mr-nr  dly

End line:              UDS/SQL Vn.n nnn  USER(S) OF confname
```

Explanations:

PROGRAM

Application program which started the transaction:

progname
Name of application program or of openUTM operand DBKEY or 'UTM'

TSN/TERM

Task identifier:

tsn

Timesharing mode: task serial number under which program *progname* runs

term-name
Transaction mode: Logical name of the openUTM client or Partner (LTERM name, LPAP name, OSI-LPAP name)

In the case of a distributed transaction, this operand refers to the primary subtransaction (PST), which may be running in another configuration.

RUNUNIT-ID

ta-id

Internal transaction identifier assigned by UDS/SQL.

Transactions are numbered consecutively within a session segment.

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

STATE

state
Indicates the present state of the transaction:

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

STOP
The transaction is in the STOP state.

AREA
Upon execution of the READY statement, the transaction is waiting in the realm queue; at present it cannot access the requested realm.

LOCK
The transaction is waiting for a locked resource such as a page to be released.

DEACT
The transaction is deactivated.

WAIT-ST
The transaction is waiting for a server task to become free. TO-STOP A STOP request is pending for the transaction.

TO-CANC
A CANCEL request is pending for the transaction.

IN-CANC
A CANCEL request is being processed for the transaction.

USER
No DML statement is currently being processed for this transaction; control resides with the application program or, in the case of UDS-D, with the UDS/SQL configuration of the PTT.

CONNEC
The transaction is under the control of the connection module or, in the case of UDS-D, of the UDS/SQL configuration of an STT.

DBH
The DBH is currently processing a DML statement for this transaction; the DBH has control.

PTC *ptc*

YES
The distributed transaction is in the Prepared to Commit (PTC) condition, i.e. the transaction FINISH procedure has been initiated but not yet completed.

-
The distributed transaction is in the Prepared to Commit (PTC) condition.

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT) or a transaction distributed via openUTM.

FC *function-code*
Function code of the DML statement which is currently being processed or which was last processed (see section “Function codes of the DML statements”)

In the case of a distributed transaction, this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

MR-NR

mainref-no

Internal number assigned by the DBH to a transaction at READY;
in contrast to *transaction-id* it is not consecutive
(value range: 1 to value of PP TRANSACTION).

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

DLY *dly*

YES

The transaction is preventing the execution of pending requests (DLY corresponds to DELAYING), i.e. it must be terminated before the requests can be executed (see the DAL commands PERFORM in chapter "Performing pending requests (PERFORM)" and RESET ORDERSi n chapter "Canceling pending requests (RESET ORDERS)").

-

The transaction is not preventing the execution of pending requests.

V*n.n* UDS/SQL version number

nnn Number of currently open transactions

DISPLAY USERS,DB=*dbname* - Show all transactions currently accessing the database

Input format: DISPLAY USERS,DB=*dbname*

Output as under output format DISPLAY USERS for application programs which have opened database *dbname* by means of READY.

Example

```
//EXECUTE-DAL-CMD CMD=DISPLAY USERS,DB=SHIPPING
```

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY USERS,DB=SHIPPING (OPOX073,09:28:53/4TMW)
4TMW: PROGRAM   TSN/TERM     RUNUNIT-ID  STATE    PTC  FC   MR-NR DLY
4TMW: ------------------------------------------------------------------
4TMW: DMLTEST   TSN 4TPM           1    USER     -    18    1    -
4TMW:
4TMW: UDS/SQL V2.9     1 USER  OF SHIPPING
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:53/4TMW)
```

DISPLAY SUBSCH - Show all subschemas

Input format: DISPLAY SUBSCH

```
Header line:                n SUBSCHEMA(S) IN confname
Information lines:
Database line:              DB = dbname
Schema line:                SCHEMA = schema-name
Subschema line(s):          subschema-name [-NAME AMBIGUOUS]
```

Explanations:

n Number of subschemas in the DB configuration

confname

First eight characters of configuration name

Database line:

dbname
Database name

Schema line:

schema-name
Schema name

Subschema line(s):

One line for each subschema of database *dbname*; it contains:

subschema-name
Subschema name (up to 30 characters)

NAME AMBIGUOUS
Addendum to all subschema names of the current DB configuration which are not unique in the first six characters. A READY to such a subschema would be rejected by the DBH with status code 141.

Example

//EXECUTE-DAL-CMD CMD=DISPLAY SUBSCH

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY SUBSCH (OPOX073,09:28:55/4TMW)
4TMW:   2 SUBSCHEMAS IN CONFEXMP
4TMW: ================================================
4TMW: DB     = SHIPPING
4TMW: SCHEMA = MAIL-ORDERS
4TMW: ------------------------------------------------
4TMW: ADMIN
4TMW: ================================================
4TMW: DB     = CUSTOMER
4TMW: SCHEMA = CUSTOMER-LIST
4TMW: ------------------------------------------------
4TMW: MANAGEMENT
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:55/4TMW)
```

DISPLAY SUBSCH,DB=*dbname* - Show all subschemas in the database

Input format: DISPLAY SUBSCH,DB=*dbname*

```
Header line:               n SUBSCHEMA(S) OF dbname
Information line(s):
Schema line:              SCHEMA = schema-name
Subschema line            subschema-name
```

n Number of subschemas in the database *dbname*

Schema line:

schema-name
Schema name

Subschema line:

subschema-name
Subschema name (up to 30 characters)

Example

//EXECUTE-DAL-CMD CMD=DISPLAY SUBSCH,DB=SHIPPING

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY SUBSCH,DB=SHIPPING (OPOX073,09:28:57/4TMW)
4TMW:  1 SUBSCHEMA  OF SHIPPING
4TMW: SCHEMA = MAIL-ORDERS
4TMW: ------------------------------------------------
4TMW: ADMIN
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:57/4TMW)
```

DISPLAY MAINREF -
Show information on mainref status

```
Comment line:        MR-NR       STATE    RUNUNIT-ID      TEMP

Information          mainref-no  state    transaction-id    temp
lines:
```

MR-NR

mainref-no

Internal number assigned by the DBH to a transaction at READY;
in contrast to the *transaction-id* it is not consecutive.
Value range: 1 to value of PP TRANSACTION

STATE

state
Indicates the current status of the mainref.

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

FREE:
The mainref is not being used by any transaction and is free for a new transaction.

STOP-NEW:
The mainref is presently locked against use by new transactions. The lock is due to:

- a DAL command, or
- a DBH-internal command

STOP
The mainref is being used by a transaction in the STOP status.

USED:
The mainref is being used by a transaction.

BLOCK:
The mainref is locked against use by new transactions for the current session section due to a defect in its temporary realm

The STATE values are not mutually exclusive. They are displayed according to the following hierarchy:

BLOCK, STOP, STOP-NEW, FREE, USED

RUNUNIT-ID

transaction-id
Internal transaction identifier assigned by UDS/SQL. Transactions are numbered consecutively within a session segment.

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

TEMP

temp
Indicates if the temporary realm is defective.

-

The temporary realm of this mainref is not defective.

DSTR
The temporary realm of this mainref has failed.

In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

If there are permanent defects in a temporary realm, the DBH responds by locking the corresponding mainref.

Example

//EXECUTE-DAL-CMD CMD=DISPLAY MAINREF

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY MAINREF (OPOX073,09:28:59/4TMW)
4TMW: MR-NR   STATE     RUNUNIT-ID   TEMP
4TMW: ---------------------------------------
4TMW:   1     USED        1          -
4TMW:   2     FREE        -          -
4TMW:   3     FREE        -          -
4TMW:   4     FREE        -          -
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:28:59/4TMW)
```

DISPLAY MAINREF, STATE=BLOCK - Show information on blocked mainrefs

Input format: `DISPLAY MAINREF,STATE=BLOCK`

```
Comment line:         MR-NR       STATE  RUNUNIT-ID      TEMP
Information line(s):  mainref-no  block  transaction-id  temp
```

If no mainref is blocked:

```
Information line: NO MAINREF IS BLOCKED
```

Explanations:

Only mainrefs locked against reuse by transactions, i.e. blocked, are listed. Otherwise, output is the same as for DISPLAY MAINREF.

Example

`//EXECUTE-DAL-CMD CMD=DISPLAY MAINREF,STATE=BLOCK`

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY MAINREF,STATE=BLOCK (OPOX073,09:28:59/4TMW)
4TMW: NO MAINREF IS BLOCKED
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:01/4TMW)
```

DISPLAY *transaction-id* - Show information on transactions

Input format: DISPLAY *transaction-id*

Comment	Information	Explanation
PROGRAM-NAME:	*progname*	Name of the application program or of the openUTM operand DBKEY or 'UTM'. In the case of a distributed transaction this operand may refer to an application program running in another configuration.
[UTM-TAC/-USER-ID:	*utm-tac/utm-userid*]	openUTM transaction code/openUTM user ID from the openUTM command KDCSIGN. In the case of a distributed transaction, this operand refers to the relevant primary subtransaction.
TSN:	*tsn*	In timesharing mode: task sequence number under which the application program runs. In the case of a distributed transaction this operand refers to the PST.
TERMINAL-NAME:	*term-name*	In transaction mode: logical name of the openUTM client or partner (LTERM name, LPAP name or OSI-LPAP name. In the case of a distributed transaction this operand refers to the PTT. The data display terminal may be attached to a different host.
RUNUNIT-ID / -STATE:	*transaction-id/* {STOP | AREA | LOCK | DEACT | WAIT-ST | TO-STOP | TO-CANC | IN-CANC | USER | CONNEC | DBH}	Transaction identifier; assigned internally by UDS/SQL. Explanation of states as for DISPLAY USERS. In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).

MAINREF-NR / - STATE:	*mainref*/ {STOP-NEW STOP | USED | BLOCK}	Mainref number; assigned when READY is issued for the transaction. In the case of a distributed transaction this operand refers to the local subtransaction, irrespective of whether it is a PTT or an STT. For STOP-NEW, STOP, BLOCK and USED, see DIS MAINREF.
FC:	*function code*	Function code of the current or most recent DML statement of the transaction (see section “Function codes of the DML statements”"). In the case of a distributed transaction, this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction (PTT or STT).
PTT-PROC-NAME:	*proc-name*	Name of the host on which the primary subtransaction is running.
PTT-CONF-NAME:	*confname*	The first 8 characters of the configuration name of the primary subtransaction.
PTT-RLOG-DATE:	*rlog-time-stamp*	Time stamp indicating the date of generation of the associated RLOG file. This operand refers to the RLOG file of the primary subtransaction.
PTT-RUNUNIT-ID:	*pst-id*	Transaction identifier of the associated PTT.
LOCAL-TT-STATE:	{PTT | STT}[IN PTC]	The local subtransaction (TT) is the PTT or STT of a distributed transaction. IN PTC: The local subtransaction is in the Prepared to Commit (PTC) condition, i.e. the interconfiguration FINISH has been initiated but not yet completed.
[UTM-D-TA:	IN PTC]	The openUTM transaction is in the PTC condition.
TEMP:	DSTR	The temporary realm of this mainref has failed. In the case of a distributed transaction, this operand refers to the local subtransaction, irrespective of whether it is a primary or secondary subtransaction.

Table 16: Input format for the output of information on transactions

Example

//EXECUTE-DAL-CMD CMD=DISPLAY 1

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY 1 (OPOX073,09:29:01/4TMW)
4TMW: PROGRAM-NAME:        DMLTEST
4TMW: TSN:                4TPM
4TMW: RUNUNIT-ID / -STATE:        1 / USER
4TMW: MAINREF-NR / -STATE:        1 / USED
4TMW: FC:                  18
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:01/4TMW)
```

DISPLAY REALMS -
Show information on all permanent realms

Input format: DISPLAY REALMS

Comment line:		DATABASE (USERID: *userid*)	ACC-USER	ACC-SYS	DB-OPT	DESTR
Information line(s):		*dbname*	*acc-user*	*acc-sys*	*db-opt*	*destr*
Comment line:	CATID	REALM	ACC-USER	ACC-SYS	STATE	DEFECT
Information line(s):	*catid*	*realm-name*	*acc-user*	*acc-sys*	*state*	*defect*

Explanations:

In certain cases realms are deactivated by the DBH. The database administrator can obtain information on the status of the realms.

userid User ID in which the database concerned is located.

DATABASE

dbname
Name of the database

CATID

catid
BS2000 catalog ID of the realm

REALM

realm-name
Name of the realm

ACC-USER

acc-user
Indicates the access mode allowed by the database administrator by means of the ACCESS command.

UPD
read and write access accepted

RTR
only read access accepted

LOCK
neither read nor write access accepted

ACC-SYS

acc-sys
Indicates the access mode allowed internally by the DBH. This mode cannot be controlled by means of the ACCESS command.

UPD
read and write access accepted

RTR
only read access accepted

LOCK
neither read nor write access accepted

DB-OPT

db-opt
Indicates the acces to the database.

EXC-UPD
EXCLUSIVE-UPDATE
The DBH accesses database *dbname* exclusively. The application programs can access the database in update mode.

SHA-RTR
SHARED-RETRIEVAL
All DBHs can have shared access to database *dbname*, i.e. this database can also be attached to other DBHs (but only in SHA-RTR mode). The application programs of all DBHs have read-only access to the database.

STATE

state
Indicates the status of the realm.

TMPRLM
The realm is temporary.

TO DROP
This realm is the subject of a detach request (DROP) which has not yet been processed (completely).

SWTOFF
The realm has been detached by the DAL command DROP RN =, the BMEND function REMOVE-REALM or DBH error handling.

TO ADD
This realm is the subject of a rejoin request which has not yet been processed (completely).

OPENED
The realm is physically open and takes part in the session.

DESTR

destr
Indicates whether or not the database is defective (DESTR=DESTROYED)

YES The database is defective
- The database is not defective

DEFECT

defect
Indicates whether the realm in the DBH is marked as defective

YES The realm is defective
- The realm is not defective

Example

//EXECUTE-DAL-CMD CMD=DISPLAY REALMS

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY REALMS (OPOX073,09:29:01/4TMW)
4TMW:        DATABASE (USERID: $XXXXXXXX.)  ACC-USER  ACC-SYS  DB-OPT  DESTR
4TMW: ------------------------------------------------------------------------
4TMW:        SHIPPING                       UPD       UPD      EXC-UPD   -
4TMW: ========================================================================
4TMW: CATID  REALM                          ACC-USER  ACC-SYS  STATE   DEFECT
4TMW: ------------------------------------------------------------------------
4TMW: :IUDS: DATABASE-DIRECTORY             UPD       UPD      OPENED    -
4TMW: :IUDS: CUSTOMER-ORDER-RLM             UPD       UPD      OPENED    -
4TMW: :IUDS: PURCHASE-ORDER-RLM             UPD       UPD      OPENED    -
4TMW: :IUDS: CLOTHING                       UPD       UPD      OPENED    -
4TMW: :IUDS: HOUSEHOLD-GOODS                UPD       UPD      OPENED    -
4TMW: :IUDS: SPORTS-ARTICLES                UPD       UPD      OPENED    -
4TMW: :IUDS: FOOD                           UPD       UPD      OPENED    -
4TMW: :IUDS: LEISURE                        UPD       UPD      OPENED    -
4TMW: :IUDS: STATIONERY                     UPD       UPD      OPENED    -
4TMW: :IUDS: ARTICLE-RLM                    UPD       UPD      OPENED    -
4TMW:   -    SEARCH-RLM                      -         -       TMPRLM    -
4TMW:
4TMW:        DATABASE (USERID: $XXXXXXXX.)  ACC-USER  ACC-SYS  DB-OPT  DESTR
4TMW: ------------------------------------------------------------------------
4TMW:        CUSTOMER                       UPD       UPD      EXC-UPD   -
4TMW: ========================================================================
4TMW: CATID  REALM                          ACC-USER  ACC-SYS  STATE   DEFECT
4TMW: ------------------------------------------------------------------------
4TMW: :IUDS: DATABASE-DIRECTORY             UPD       UPD      OPENED    -
4TMW: :IUDS: CUSTOMER-RLM                   UPD       UPD      OPENED    -
4TMW: :IUDS: FINANCE-RLM                    UPD       UPD      OPENED    -
4TMW
\
:   -         TEMPRLM                     -         -       TMPRLM    -
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:03/4TMW)
```

DISPLAY REALMS,DB=*dbname* - Show information on all permanent realms of a database

Input format: DISPLAY REALMS,DB=*dbname*

Explanation: Same as DISPLAY REALMS; output refers to the database *dbname* only

Example

//EXECUTE-DAL-CMD CMD=DISPLAY REALMS,DB=SHIPPING

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY REALMS,DB=SHIPPING (OPOX073,09:29:07/4TMW)
4TMW:        DATABASE (USERID: $XXXXXXXX.)  ACC-USER  ACC-SYS  DB-OPT  DESTR
4TMW: ------------------------------------------------------------------------
4TMW:        SHIPPING                       UPD       UPD      EXC-UPD   -
4TMW: ========================================================================
4TMW: CATID  REALM                          ACC-USER  ACC-SYS  STATE   DEFECT
4TMW: ------------------------------------------------------------------------
4TMW: :IUDS: DATABASE-DIRECTORY             UPD       UPD      OPENED    -
4TMW: :IUDS: CUSTOMER-ORDER-RLM             UPD       UPD      OPENED    -
4TMW: :IUDS: PURCHASE-ORDER-RLM             UPD       UPD      OPENED    -
4TMW: :IUDS: CLOTHING                       UPD       UPD      OPENED    -
4TMW: :IUDS: HOUSEHOLD-GOODS                UPD       UPD      OPENED    -
4TMW: :IUDS: SPORTS-ARTICLES                UPD       UPD      OPENED    -
4TMW: :IUDS: FOOD                           UPD       UPD      OPENED    -
4TMW: :IUDS: LEISURE                        UPD       UPD      OPENED    -
4TMW: :IUDS: STATIONERY                     UPD       UPD      OPENED    -
4TMW: :IUDS: ARTICLE-RLM                    UPD       UPD      OPENED    -
4TMW:   -    SEARCH-RLM                      -         -       TMPRLM    -
4TMW:
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:09/4TMW)
```

DISPLAY REALMS,RN=*realm-name*[,DB=*dbname*] - Show information on a permanent realm

Input format: DISPLAY REALMS,RN=*realm-name*[,DB=*dbname*]

Explanation: Same as DISPLAY REALMS; output refers to the realm *realm-name* [of database *dbname*] only

The specification of DB=*dbname* can be omitted if the realm name specified is unambiguous in the configuration.

DISPLAY PP - Show current DBH load parameters

Input format: DISPLAY PP

Information line(s):

```
MAXDB            = n 1

TRANSACTION      = n 2 or (n 2,n 3) for UDS-D

[SUBTRANSACTION  = n 4]

SUBSCHEM         = n 5

SERVERTASK       = n 6

2KB-BUFFER-      = n 7a
SIZE

4KB-BUFFER-      = n 7b
SIZE

8KB-BUFFER-      = n 7c
SIZE

CP-SIZE          = n 8

CUP-SIZE         = n 9

SIP-SIZE         = n 10

[DIP-SIZE        = n 11] for UDS-D

[DISDB           = n 12] for UDS-D

[CHCKTIME        = n 13] for UDS-D

DEADTIME         = n 14

DISTRIBUTION     = {NO | STANDBY | START}

PTCSYNCH         = ({WAIT | ABORT | COMMIT})

LOG              = {NO |:catid:|PUBLIC |
                    (priv-vsn-1/device-1[,priv-vsn-2
                   device-2
                   [,priv-vsn-3 device-3]]) |
                   (vsn-1[,vsn-2[,vsn-3]])}
```

```
LOG-2           = {NO |
                   :catid: |
                   PUBLIC |
                   (priv-vsn-1/device-1[,priv-vsn-2
                  device-2
                  [, priv-vsn-3 device-3 ]]) |
                  (vsn-1[,vsn-2[,vsn-3]])}

LOG-SIZE        = (primary,secondary])

RESERVE         = {NONE | :catid: |
                  (priv-vsn-1/gerät-1[,priv-vsn-2 gät-2
                  [,priv-vsn-3 gerät-3]]) |
                  (vsn-1[,vsn-2[,vsn-3]])}

WARMSTART       = {STD | FAST | VERY-FAST}

DEACT           = {YES | NO}

STDCKPT         = {YES | NO}

ADM             = {LOCAL | REMOTE}

CP              = {MONO-PROCESSOR | MULTI-PROCESSOR}

[SQL            = n 15]

[SQL-LIMIT      = n 16]

[DISTABLE       = {NO | [:catid:][$userid.]file-name}]
                  for UDS-D

TASKLIB         = {NO | [:catid:][$userid.]udsmodlib}

DUMP            = {STD | ALL}

MPSEG           = {STD | 64K}

UCON            = C'{(mn) | <x | nnnn}]'[,{MSG | UDS}]

SECLEVEL        = {F2 | F2-EXCEPT | NO},NO-AUDIT

LOCK            = {STD | SHARED | EXCLUSIVE}

TA-ACCESS       = {STD | SHARED}

WAIT            = {EVENT | BUSY}

RESULT-DELAY    = n 17

SCHEDULING      = {SYMMETRIC | ASYMMETRIC}

IO              = {ASYNC | SYNC}
```

```
ORDER-DBSTATUS   = {STD | SPECIAL}

PRIVACY-CHECK    = {STD | NO-KSET | OFF}

BCAM-PREFIX      = prefix

CONFNAME         = [$ userid. ] confname

DATABASES OF CONFIGURATION:

$userid.dbname,{EXCLUSIVE-UPD
[,bbbb][,buffid|SHARED-RTR[,bbbb][,buffid}

                Up to MAXDB lines; empty configuration
```

n_1 Number of databases in the DB configuration

n_2 Max. number of simultaneously active transactions

n_3 Max. number of transactions simultaneously active for remote application programs

n_4 Number of KDBS logical file names (KDBS only)

n_5 Max. number of subschemas usable simultaneously per database

n_6 Number of server tasks

n_{7a} Size of the system buffer pool for databases with a 2-Kbyte page format in Mbytes

n_{7b} Size of the system buffer pool for databases with a 4-Kbyte page format in Mbytes

n_{7c} Size of the system buffer pool for databases with an 8-Kbyte page format in Mbytes

n_8 Minimum size of a common pool in Kbytes

n_9 Size of the communication pool in Kbytes

n_{10} Size of the SSITAB pool in Kbytes

n_{11} Size of the distribution pool in Kbytes

n_{12} Max. number of remote databases per transaction

n_{13} Time, in seconds, for connection and transaction checking by the UDS-D task

n_{14} Time, in seconds, for resolving interconfiguration deadlocks or deadlocks involving openUTM

n_{15} Number of simultaneously active SQL conversations

n_{16} Time for inactive conversations

n_{17} Number of request responses to be grouped

bbbb Size of the user buffer pool of the database in Mbytes without leading zeros; omitted if the system buffer pool has been assigned to the database.

buffid Name of the shared user buffer pool, if a shared user buffer pool has been assigned to the database.
*EXCL indicates that an exclusive user buffer pool has been assigned to the database.
*SYSTEM indicates that the system buffer pool has been assigned to the database.

prefix The BCAM names of the user tasks are formed from *prefix* and TSN.

Explanations:

The call outputs the current values of the DBH load parameters, including DAL changes.

They are listed in the same form as when PP PARLIST=YES is specified during the start phase.

The load parameters TASKLIB and SECLEVEL are ineffective. They are only still supported for compatibility reasons.

Example

//EXECUTE-DAL-CMD CMD=DISPLAY PP

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY PP (OPOX073,09:29:11/4TMW)
4TMW: MAXDB          =      2
4TMW: TRANSACTION    = (    4,    1)
4TMW: SUBSCHEMA      =      1
4TMW: SERVERTASK     =      1
4TMW: 2KB-BUFFER-SIZE=      1
4TMW: 4KB-BUFFER-SIZE=      1
4TMW: 8KB-BUFFER-SIZE=      0
4TMW: CP-SIZE        =   1024
4TMW: CUP-SIZE       =   1024
4TMW: SIP-SIZE       =   1024
4TMW: DIP-SIZE       =   1024
4TMW: DISDB          =      1
4TMW: CHCKTIME       =     60
4TMW: DEADTIME       =     60
4TMW: DISTRIBUTION   = START
4TMW: PTCSYNCH       = (WAIT  ,WAIT  )
4TMW: LOG            = PUBLIC
4TMW: LOG-2          = NO
4TMW: LOG-SIZE       = (    192,    192)
4TMW: RESERVE        = NONE
4TMW: WARMSTART      = STD
4TMW: DBDCSYNCH      = YES
4TMW: DEACT          = YES
4TMW: STDCKPT        = YES
4TMW: ADM            = REMOTE
4TMW: CPU            = MONO-PROCESSOR
4TMW: DISTABLE       = UDSDBB.VT.EXAMPLE.START
4TMW: SQL            =     4
4TMW: SQL-LIMIT      =     10
4TMW: TASKLIB        = NO
4TMW: DUMP           = ALL
4TMW: MPSEG          = STD
4TMW: UCON           = C'<U  ',MSG
4TMW: SECLEVEL       = NO      ,NO-AUDIT
4TMW: LOCK           = STD
4TMW: TA-ACCESS      = STD
4TMW: WAIT           = EVENT
4TMW: RESULT-DELAY   =      0
4TMW: SCHEDULING     = SYMMETRIC
4TMW: IO             = ASYNC
4TMW: ORDER-DBSTATUS = STD
4TMW: PRIVACY-CHECK  = OFF
4TMW: BCAM-PREFIX    = SUD$
4TMW: CONFNAME       = $XXXXXXXX.CONFEXMP
4TMW: DATABASES OF CONFIGURATION:
4TMW:   $XXXXXXXX.SHIPPING                    ,EXCLUSIVE-UPD     ,*SYSTEM
4TMW:   $XXXXXXXX.CUSTOMER                    ,EXCLUSIVE-UPD,   1,*EXCL
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  COMMAND  (OPCC074,09:29:13/4TMW)
```

DISPLAY FPA -
Show information on free pages in all realms

Input format: DISPLAY FPA

Comment line: DATABASE

Information line(s): *dbname*

Comment lines:	RLM REF	REALM	NR PAGES	NR FREE PAGES	SECOND SCAN
Information line(s):	*realm-reference*	*realm-name*	*n*	*n*	*second scan*

Explanations:

The database administrator can obtain information on the number of free pages in the realms.

DATABASE

dbname
Name of the database

RLMREF

realm-reference
Number of the realm

REALM

realm-name
Name of the realm

NR PAGES *n*

Number of pages in the realm

NR FREE PAGES *n*

Number of free pages in the realm

SECOND SCAN -

second scan
Search mode

-
The realm has not yet been searched completely for free pages. The search mode"First Scan" is used (space for new data which is to be stored is searched for in the free area at the end of the realm).

YES
The realm has already been searched completely for free pages once. The search mode "Second Scan" is used (space for new data which is to be stored is searched for from the start of the realm).

Example

//EXECUTE-DAL-CMD CMD=DISPLAY FPA

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY FPA (OPOX073,09:29:22/4TMW)
4TMW: DATABASE
4TMW: ------------------------------------------------------------
4TMW: SHIPPING
4TMW: ============================================================
4TMW: RLM  REALM                          NR      NR FREE  SECOND
4TMW: REF                                 PAGES   PAGES    SCAN
4TMW: ------------------------------------------------------------
4TMW:   1  DATABASE-DIRECTORY               100       49    -
4TMW:   3  CUSTOMER-ORDER-RLM                36       22    -
4TMW:   4  PURCHASE-ORDER-RLM                60       33    -
4TMW:   5  CLOTHING                          54       29    -
4TMW:   6  HOUSEHOLD-GOODS                   24       16    -
4TMW:   7  SPORTS-ARTICLES                   44       35    -
4TMW:   8  FOOD                              18        9    -
4TMW:   9  LEISURE                           44       35    -
4TMW:  10  STATIONERY                        24       17    -
4TMW:  11  ARTICLE-RLM                      62       13    -
4TMW:
4TMW: DATABASE
4TMW: ------------------------------------------------------------
4TMW: CUSTOMER
4TMW: ============================================================
4TMW: RLM  REALM                          NR      NR FREE  SECOND
4TMW: REF                                 PAGES   PAGES    SCAN
4TMW: ------------------------------------------------------------
4TMW:   1  DATABASE-DIRECTORY               100       55    -
4TMW:   3  CUSTOMER-RLM                     126       23    -
4TMW:   4  FINANCE-RLM                      126      117    -
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:22/4TMW)
```

DISPLAY FPA,DB=*dbname* -
Show information on free pages in all realms of a database

Input format: DISPLAY FPA,DB=*dbname*

Explanation: Same as DISPLAY FPA; output refers to database *dbname* only

DISPLAY FPA,RN=*realm-name*[,DB=*dbname*]-
Show information on free pages in a realm

Input format: DISPLAY FPA,RN=*realm-name*[,DB=*dbname*]

Explanation: Same as DISPLAY FPA; output refers to the realm *realm-name* [of database *dbname*] only

The specification of DB=*dbname* can be omitted if the realm name specified is unambiguous in the configuration.

DISPLAY XFPA -
Show extended information on free pages in all realms

Input format: DISPLAY XFPA

Comment line: DATABASE

Information line(s) *dbname*

```
Comment       RLM      NR    NR FREE NR FULL PARTIAL-FILLED  SECOND SEARCH
lines:        REF      PAGES PAGES   PAGES   NR PAGES  FIL%  SCAN   IN FPA

Information   realm-   n     n       n       n         n           second search
line(s):      reference                                            scan   in fpa
```

Explanations:

The database administrator can obtain information on the number of (free) pages in the realms.

DATABASE *dbname*
Name of the database

RLMREF *realm-reference*
Number of the realm

NR PAGES

Number of pages in the realm

NR FREE PAGES

Number of free pages in the realm

NR FULL PAGES

Number of full pages in the realm

PARTIAL-FILLED NR PAGES

Number of partially filled pages in the realm

FIL% Average filling level of all partially filled pages in the realm

SECOND SCAN

second scan
Search mode

-
The realm has not yet been searched completely for free pages. The search mode"First Scan" is used (space for new data which is to be stored is searched for in the free area at the end of the realm).

YES
The realm has already been searched completely for free pages once. The search mode "Second Scan" is used (space for new data which is to be stored is searched for from the start of the realm).

SEARCH IN FPA

search in fpa
The value of SET-FPA-SCAN-PARAMETERS which was stored by the UDS online utility or, for BMODTT, the value SET/RESET REUSE-FREE-SPACE.

REUSE
Space for new records and tables which are to be stored is searched for from the start of the realm (Second Scan).

NOREUSE
Space for new records and tables which are to be stored is searched for at the end of the realm after the first page of the contiguous range of free pages (First Scan).

Example

//EXECUTE-DAL-CMD CMD=DISPLAY XFPA

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY XFPA (OPOX073,09:29:28/4TMW)
4TMW: DATABASE
4TMW: ----------------------------------------------------------------
4TMW: SHIPPING
4TMW: ================================================================
4TMW: RLM     NR      NR FREE   NR FULL  PARTIAL-FILLED  SECOND  SEARCH
4TMW: REF     PAGES   PAGES     PAGES    NR PAGES  FIL%  SCAN    IN FPA
4TMW: ----------------------------------------------------------------
4TMW:   1       100        49        44         7    75    -     NOREUSE
4TMW:   3        36        22        13         1     2    -     NOREUSE
4TMW:   4        60        33        25         2    37    -     NOREUSE
4TMW:   5        54        29        10        15    27    -     NOREUSE
4TMW:   6        24        16         8         0     -    -     NOREUSE
4TMW:   7        44        35         9         0     -    -     NOREUSE
4TMW:   8        18         9         5         4     7    -     NOREUSE
4TMW:   9        44        35         9         0     -    -     NOREUSE
4TMW:  10        24        17         7         0     -    -     NOREUSE
4TMW:  11        62        13        42         7     7    -     NOREUSE
4TMW:
4TMW: DATABASE
4TMW: ----------------------------------------------------------------
4TMW: CUSTOMER
4TMW: ================================================================
4TMW: RLM     NR      NR FREE   NR FULL  PARTIAL-FILLED  SECOND  SEARCH
4TMW: REF     PAGES   PAGES     PAGES    NR PAGES  FIL%  SCAN    IN FPA
4TMW: ----------------------------------------------------------------
4TMW:   1       100        55        38         7    74    -     NOREUSE
4TMW:   3       126        23       100         3    39    -     NOREUSE
4TMW:   4       126       117         7         2    22    -     NOREUSE
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:28/4TMW)
```

DISPLAY XFPA,DB=*dbname* - Show extended information on free pages in all realms of a database

Input format: DISPLAY XFPA,DB=*dbname*

Explanation: Same as DISPLAY XFPA; output refers only to the database specified with *dbname*.

DISPLAY XFPA,RN=*realm-name*[,DB=*dbname*] - Show extended information on free pages in a realm

Input format: DISPLAY XFPA,RN=*realm-name*[,DB=*dbname*]

Explanation: Same as DISPLAY XFPA; output refers only to the realm specified with *realm-name* [of the database *dbname*].

The specification of DB=*dbname* can be omitted if the realm name specified is unambiguous in the configuration.

DISPLAY INCR - Show information on online extensibility of all activated realms in the database configuration

Input format: DISPLAY INCR

```
Information line(s):   DATABASE (USERID: $xxxxxxxx.)
Information line(s):   dbname
Comment lines:         RLM REALM       INCR  NR      MIN      FMT   EXT  EXT
                       REF             ACT      PAGES   PAGES       PND  SIZE
Information line(s):   n   realmname   incr  n       n        fmt   ext  n
                                       act                         pnd
```

Explanations:

The database administrator can obtain information on the online extensibility of the realms.

USERID

User ID in which the database concerned is located.

DATABASE

dbname
Name of the database

RLM REF

realmname
Number of the realm

REALM

Name of the realm

INCR ACT

incr act
Online extensibility

YES
Online extensibility is activated for the realm and not suspended.

NO
Online extensibility is not activated for the realm.

SUSP
Online extensibility is activated for the realm, but suspended in the current part of the session.

NR PAGES *n*

Number of pages by which the realm is to be extended if required.

MIN PAGES *n*

Number of free pages in the realm below which an online realm extension will be started.

FMT

fmt
Formatierung der neu hinzugefügten Datenbankseiten

YES
It is necessary for the DBH to format the newly added database pages.

NO
It is not necessary for the DBH to format the newly added database pages.

EXT PND

ext pnd
Display for unconditional realm extension.

YES
Unconditional one-time realm extension (one realm extension per DAL command) is noted.

NO
Unconditional one-time realm extension (one realm extension per DAL command) is not noted.

EXT SIZE

Number of database pages required for extension.
This is displayed only if unconditional one-off realm extension is noted, in other words EXT PND is equal to YES.

Example

//EXECUTE-DAL-CMD CMD=DISPLAY INCR

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY INCR (OPOX073,09:29:34/4TMW)
4TMW: DATABASE (USERID: $XXXXXXXX.)
4TMW: ------------------------------------------------------------------------
4TMW: SHIPPING
4TMW: ========================================================================
4TMW: RLM REALM                       INCR NR      MIN       FMT EXT   EXT
4TMW: REF                             ACT     PAGES    PAGES     PND   SIZE
4TMW: ------------------------------------------------------------------------
4TMW:   1 DATABASE-DIRECTORY          YES        64        0 YES NO
4TMW:   3 CUSTOMER-ORDER-RLM          YES      2000      400 NO  NO
4TMW:   4 PURCHASE-ORDER-RLM          YES        64        0 NO  NO
4TMW:   5 CLOTHING                    YES        64        0 NO NO
4TMW:   6 HOUSEHOLD-GOODS             YES        64        0 NO  NO
4TMW:   7 SPORTS-ARTICLES             YES        64        0 NO  NO
4TMW:   8 FOOD                        YES       500       50 NO  NO
4TMW:   9 LEISURE                     YES        64        0 NO  NO
4TMW:  10 STATIONERY                  YES        64        0 NO  NO
4TMW:  11 ARTICLE-RLM                 YES      1000      100 NO  NO
4TMW:
4TMW: DATABASE (USERID: $XXXXXXXX.)
4TMW: ------------------------------------------------------------------------
4TMW: CUSTOMER
4TMW: ========================================================================
4TMW: RLM REALM                       INCR NR      MIN       FMT EXT   EXT
4TMW: REF                             ACT     PAGES    PAGES     PND   SIZE
4TMW: ------------------------------------------------------------------------
4TMW:   1 DATABASE-DIRECTORY          YES        64        0 YES NO
4TMW:   3 CUSTOMER-RLM                YES       500       30 NO  NO
4TMW:   4 FINANCE-RLM                 YES        64        0 NO  NO
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:34/4TMW)
```

DISPLAY INCR, DB=*dbname* - Show information on the online extensibility of all activated realms in a database

Input format: DISPLAY INCR, DB=*dbname*

Explanation: As for DISPLAY INCR; the output only relates to the active realms of the database *dbname*.

DISPLAY INCR,RN=*realm-name* [,DB=*dbname*] - Show information on the online extensibility of a realm

Input format: DISPLAY INCR[, DB=*dbname*],RN=*realm-name*

Explanation: As for DISPLAY INCR; the output only relates to the realm *realm-name*.

The specification DB=*dbname* can be omitted if the named realm name is unique in the configuration.

DISPLAY DBTT-INCR -
Show information on the online DBTT extensibility of the record types for the user schemata in the DB configuration

Input format: DISPLAY DBTT-INCR

```
Information line(s):  DATABASE (USERID: $xxxxxxxx.)

Information line(s):  dbname

Comment lines:        REC RECORD    RLM DBTT   NR      EXT   HIGHEST    EXT
                      REF           REF ACT    ENTRIES PND          RSQ SIZE

Information line(s):  n   recordname n  dbtt   n       ext   n          n
                                        act            pnd
```

Explanations:

The database administrator can obtain information on the online extensibility of the DBTTs.

USERID

User ID in which the database concerned is located.

dbname

Name of the database

REC REF *n*

Number of the record type

RECORD

recordname
Name of the record type

RLM REF *n*

Number of the realm
The DAL command DISPLAY INCR,DB=*dbname* can be used to identify the name of the realm.

DBTT ACT

dbtt act
Indicates whether online DBTT extension is active:

SCAN
Online DBTT extension is active and not suspended. No online DBTT extension is performed until no further free entry can be found in the entire DBTT.

NOSCAN
Online DBTT extension is active and not suspended. Online DBTT extension is performed if the search for a free entry is unsuccessful with the highest currently possible RSQ. Consequently, no renewed search for free entries with a lower RSQ is undertaken.

NO
Online DBTT extension is not active.

SUSP
Online DBTT extension has been suspended in the current session.
This may be because online realm extension has been suspended in the corresponding realm or because the maximum size of the DBTT has been reached.

SWTOFF
The realm containing the DBTT is not currently available. The activation state is unclear.

NR ENTRIES *n*

Number of entries for DBTT extension.

EXT PND

ext pnd
Information on as yet unexecuted EXTEND-DBTT requests:

YES
A request has been noted for this record type. The request will be executed the next time a record is stored.

NO
No request has been noted.

HIGHEST RSQ *n*

Highest possible RSQ for the record type in the current DBTT

EXT SIZE *n*

Number of DBTT entries required for extension.
This is displayed only if unconditional one-off DBTT extension is noted, in other words EXT PND is equal to YES.

If the maximum number of output lines for a DISPLAY command is not sufficient to output the information for all the record types then a corresponding message is output:

```
OUTPUT OF nnnn FURTHER LINES SUPPRESSED
```

There is no command allowing you to resume this interrupted DISPLAY command. To obtain the information for all the record types, you must repeat the command, for example with LINES=ALL. The corresponding information can also be provided for individual databases using the BSTATUS utility routine.

In all cases, the following message is then output:

```
UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND
```

If an online DBTT extension fails, you can obtain information on the reasons for its suspension as follows:

- with the DAL command DISPLAY INCR
- using the value HIGHEST RSQ in the DISPLAY DBTT-INCR command

Example

//EXECUTE-DAL-CMD CMD=DISPLAY DBTT-INCR

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY DBTT-INCR (OPOX073,09:29:40/4TMW)
4TMW: DATABASE (USERID: $XXXXXXXX.)
4TMW: ------------------------------------------------------------------------
4TMW: SHIPPING
4TMW: ========================================================================
4TMW: REC   RECORD               RLM DBTT   NR       EXT HIGHEST    EXT
4TMW: REF                        REF ACT    ENTRIES  PND        RSQ SIZE
4TMW: ------------------------------------------------------------------------
4TMW:     2 CUSTOMER               3 SCAN      21184 NO        331
4TMW:     3 CST-ORDERS             3 SCAN      31808 NO        497
4TMW:     4 ORD-ITEM               3 SCAN      63680 YES      1990    63680
4TMW:     5 INSTALMENT             3 SCAN      63680 NO        995
4TMW:     6 ART-TYPE              11 NOSCAN    31808 NO        497
4TMW:     7 ART-SELECTION         11 SCAN      31808 NO        497
4TMW:     8 ART-DESCR             11 SCAN      31808 NO        497
4TMW:     9 ARTICLE               11 NOSCAN    63680 YES       995    63680
4TMW:    10 SUBSET                 6 SCAN      63680 NO        995
4TMW:    11 COLORS                11 SCAN      63680 NO        995
4TMW:    12 MATERIALS             11 NOSCAN    63680 NO        995
4TMW:    13 SUPPLIER               4 SCAN      21184 NO        662
4TMW:    14 PURCHASE-ORDER         4 SCAN      31808 NO        497
4TMW:    15 P-ORD-ITEM             4 SCAN      63680 NO        995
4TMW:
4TMW: DATABASE (USERID: $XXXXXXXX.)
4TMW: ------------------------------------------------------------------------
4TMW: CUSTOMER
4TMW: ========================================================================
4TMW: REC   RECORD              RLM DBTT   NR        EXT HIGHEST    EXT
4TMW: REF                       REF ACT    ENTRIES   PND       RSQ SIZE
4TMW: ------------------------------------------------------------------------
4TMW:     2 CUSTOMER              3 SCAN      21184  NO        662
4TMW:     3 OPEN-ITEMS            3 SCAN      63680  NO        995
4TMW:     4 KREDIT                4 SCAN      63680  NO       1990
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:29:40/4TMW)
```

DISPLAY DBTT-INCR,DB=*dbname* - Show information on the online DBTT extensibility of the record types for the user schema in a database

Input format: DISPLAY DBTT-INCR,DB=*dbname*

Explanation: As DISPLAY DBTT-INCR; the output relates only to the record types of the user schema in the database *dbname*.

DISPLAY DBTT-INCR,DB=*dbname*,RECR=*recordref* - Show information on the online DBTT extensibility of a record type

Input format: DISPLAY DBTT-INCR,DB=*dbname*,RECR=*recordref*

Explanation: As DISPLAY DBTT-INCR; the output relates only to the record type with the record type *recordref* of the user schema in the database *dbname*.

DISPLAY PUBSETS -
Show the settings of the UDS/SQL pubset declaration

Input format: DISPLAY PUBSETS

Explanations:

`Header line:`	`UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS`
`Information line:`	`UDS-PUBSET-JV: `*`jv-name`*
`Information line(s):`	`PUBSETS: `*`catid-group`*
`End line:`	`DEFAULT PUBSET: `*`default-catid`*

jv-name

Name of the pubset declaration job variable

catid-group

Catid group which was specified in the UDS/SQL pubset declaration.

default-catid

Default catalog ID of the configuration user ID

The database administrator can obtain information on the current settings of the UDS/SQL pubset declaration. Between the DAL commands NEW PUBSETS and PERFORM the settings of the new UDS pubset declaration are also shown in the same format but with the insert "PENDING" instead of "CURRENT" in message UDS0746.

If no UDS pubset declaration is available, the following is shown to indicate that there is no explicitly specified catid group "*":

```
UDS0747 UDS PUBSET DECLARATION NOT PRESENT, DEFAULT IS CATID * (CURRENT) (...,tsn4)
```

DISPLAY ALOG [,DB=*dbname*] - Show ALOG settings

Input format: DISPLAY ALOG [,DB=*dbname*]

```
Comment line: USERID  DATABASE-NAME DEF-SUP  RES-SUP  PRIM-ALL SEC-ALL SHARE

Information   userid .dbname        def-cat  res-cat  prim     sec     share
line(s):


End line:     UDS/SQL Vn.n nnn DATABASE(S) IN confname
```

Explanations:

The database administrator can obtain information on settings of the ALOG files.

USERID

userid
User identification to which the database is assigned.

DATABASE-NAME

dbname
Name of the database

DEF-SUP

def-cat
BS2000 catalog ID for default support

RES-SUP

res-cat
BS2000 catalog ID for reserve support

PRIM-ALL

prim
Primary allocation of ALOG file

SEC-ALL

sec
Secondary allocation of ALOG file

SHARE

share
ALOG file accessibility

YES
ALOG file is accessible for all users

NO
ALOG file is accessible only for owner

V*n.n* UDS/SQL version number

nnn Number of databases in the DB configuration

Example

```
0L9M: USERID    DATABASE-NAME       DEF-SUP  RES-SUP  PRIM-ALL  SEC-ALL  SHARE
0L9M: ------------------------------------------------------------------------
0L9M: $XXXXXXXX.SHIPPING               IUDS     ABN2       192      576     NO
0L9M: $XXXXXXXX.CUSTOMER               IUDS     IUDS       300      600    YES
0L9M:
0L9M: UDS/SQL V2.9   2 DATABASES IN CONFEXMP
```

4.4.15 Displaying all or part of the distribution table (&DISPLAY DISTRIBUTION)

For UDS-D

```
&DISPLAY DISTRIBUTION
  [,NODE=processor-name]
  [,CONF=confname]
  [,DB=dbname]
  [,SS=subschema-name]
```

processor-name

Name of a host; must be no more than eight characters in length.

confname

The first eight characters of *configuration-name*; must be unique in its first seven characters. The eighth character must not be '@'. Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

dbname

Name of a database;
must be unique network-wide.

The specification :*catid*:$*userid*.*dbname*.*realm-name*.*copy-name* may be at most 54 characters in length.

subschema-name

Name of a subschema; may be at most 30 characters in length; must be unique network-wide in its first six characters.

> Locked entries are identified with a prefixed minus sign.
> Distribution table output is limited to 32 KB. If the distribution table is larger than this then it is not output in full and a message is issued to inform you of this.

UDS-D displays the following information on the display terminal:

If &DISPLAY DISTRIBUTION is specified without operands, the entire local distribution table is shown:

Example 1

//EXECUTE-DAL-CMD CMD=&DISPLAY DISTRIBUTION

```
%  UDS0220 UDS RECEIVED COMMAND: &DISPLAY DISTRIBUTION (OPCC074,09:29:40/4TMW)
4TMW: ================================================================
4TMW:  NODE-NAME  CONF-NAME  DB-NAME                       SS-NAME
4TMW: ================================================================
4TMW:   PROZ1      CONF3      DB11                          SS111
4TMW:                                                      -SS112
4TMW:                        ----------------------------------------
4TMW:                         DB12                          SS121
4TMW:                                                       SS122
4TMW: ----------------------------------------------------------------
4TMW:   PROZ2      CONF4      DB21                          SS21
4TMW:                        ----------------------------------------
4TMW:                         DB22                          SS22
4TMW: ----------------------------------------------------------------
%  UDS0832 UDS-D: COMMAND EXECUTED  (OPCC382,09:29:48/4TMW)
```

If &DISPLAY DISTRIBUTION is specified with one operand, the given operand and its immediate neighbors in the distribution table are shown:

Example 2

//EXECUTE-DAL-CMD CMD=&DISPLAY DISTRIBUTION,DB=DB11

```
%  UDS0220 UDS RECEIVED COMMAND: &DISPLAY DISTRIBUTION,DB=DB11 (OPOX073,09:29:50/4TMW)
4TMW: ================================================================
4TMW:  NODE-NAME  CONF-NAME  DB-NAME                       SS-NAME
4TMW: ================================================================
4TMW:             CONF3      DB11                          SS111
4TMW:                                                     -SS112

4TMW: ----------------------------------------------------------------
%  UDS0832 UDS-D: COMMAND EXECUTED  (OPCC382,09:29:50/4TMW)
```

If &DISPLAY DISTRIBUTION is specified with more than one operand, the given operands and their immediate neighbors in the distribution table are shown:

Example 3

//EXECUTE-DAL-CMD CMD=&DISPLAY DISTRIBUTION,CONF=CONF3,DB=DB11

```
%  UDS0220 UDS RECEIVED COMMAND: &DISPLAY DISTRIBUTION,CONF=CONF3,DB=DB11 (OPOX073,09:29:52
/4TMW)
4TMW: ================================================================
4TMW:  NODE-NAME  CONF-NAME  DB-NAME                      SS-NAME
4TMW: ================================================================
4TMW:   PROZ1      CONF3     DB11                          SS111
4TMW:                                                    -SS112

0YBI: ----------------------------------------------------------------
%  UDS0832 UDS-D: COMMAND EXECUTED  (OPCC382,09:29:52/4TMW)
```

4.4.16 Displaying information on an SQL conversation (DISPLAY SQL)

```
DISPLAY SQL{,VG=con-no |
           ,OPTION=ALL[,VG=con-no] |

           ,OPTION=IDLE[,TIME=t]}
```

con-no Number of the conversation;

number of the SQL conversation, assigned internally by UDS/SQL; conversations are numbered consecutively during each session segment.

t Time interval from 1..999 minutes

The DISPLAY SQL command logs information to the database administrator’s terminal or to the operator console as follows:

VG=*con-no*

all information for conversation *con-no*

OPTION=ALL

all conversations, sorted in order of increasing age

OPTION=ALL,VG=*con-no*

all conversations, beginning with the conversation specified, sorted in order of increasing age

OPTION=IDLE

the conversations inactive for the longest amount of time, sorted in order of increasing duration of inactivity

OPTION=IDLE,TIME=*t*

all inactive conversations that have been inactive at least *t* minutes, sorted in order of increasing duration of inactivity

The following sections describe the output of the various DISPLAY SQL commands.

Show all information for a conversation

Input format: DISPLAY SQL,VG=*con-no*

```
Information lines:  VG-NR:                 con-no
                    UTM-APPL-/USER-NAME:  appl-/user-name
                    BATCH/TSN:            tsn
                    VG-START:             date
                    NR-TA:                no-ta
                    NR-STMT:              no-stmt
                    MEM:                  mem (Kbyte)
                    SQL-OPC:              opcode
                    TA-ID:                ta-id
                    IDLE-TIME:            time (minutes)
```

Explanations:

con-no

Conversation number; up to eight digits;

number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each part of a session.

appl-/user-name

openUTM application name/user name; up to 17 characters

tsn Task serial number under which the application program is running; up to four characters

date Time stamp for the beginning of the conversation in UDS/SQL: *yyyy-mm-dd hh*:*mm*:*ss*

no-ta Number of completed SQL transactions in the conversation; up to four digits

no-stmt

Number of SQL statements processed in the conversation; up to four digits

mem Amount of SQL-specific memory, in Kbyte; up to four digits

opcode Current or most recent SQLU statement code; up to six characters, as follows:

```
PERMIT      PERMIT
CLOCUR      CLOSE CURSOR
COMWOR      COMMIT WORK
CRETV       CREATE TEMPORARY VIEW
DCLCUR      DECLARE CURSOR
DELPOS      DELETE POSITIONED
DELSRC      DELETE SEARCHED
FETCH       FETCH
INSERT      INSERT
OPECUR      OPEN CURSOR
RESCUR      RESTORE CURSOR
ROLWOR      ROLLBACK WORK
SELECT      SELECT
SETTA       SET TRANSACTION
STOCUR      STORE CURSOR
UPDPOS      UPDATE POSITIONED
UPDSRC      UPDATE SEARCHED
```

ta-id Global transaction ID of the currently open TA in the conversation; up to eight characters, or "- - -" if no TA is open or has as yet been opened due to the DAL command STOP NEW.

time Duration of inactivity, in minutes; up to four digits, or "- - -" if a TA is open, i.e. the conversation is active.

If the amount of space for a particular output field is insufficient to display the current value, the string ">>...>" is displayed instead.

Show all active and inactive conversations

Input format: DISPLAY SQL,OPTION=ALL

```
Header line:            VG-NR    TYPE    VG-START          MEM  TA-ID    IDLT
Information line(s):    con-no   type    date              mem  ta-id    time
End line:  {END OF DISPLAY | CONTINUE FOR n FURTHER VGS}
```

Explanations:

Conversation number; up to eight digits;

con-no Conversation number; up to eight digits;

number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each session segment.

type Type of application which initiated the operation:

UTM
A UDS/SQL-openUTM application

BATCH
A UDS/SQL-TIAM application

date Time stamp for the beginning of the conversation in UDS/SQL:
yyyy-mm-dd hh:*mm*:*ss*

mem Amount of SQL-specific memory, in Kbytes; up to four digits

ta-id Global transaction ID of the currently open TA in the conversation; up to eight characters, or "- - -" if no TA is open or has as yet been opened due to the DAL command STOP NEW.

time Duration of inactivity, in minutes; up to four digits, or "- - -" if a TA is open, i.e. the conversation is active

n Number of conversations not yet shown; up to four digits

If the amount of space for a particular output field is insufficient to display the current value, the string ">>...>" is displayed instead.

All active conversations are shown, sorted in order of increasing age, beginning with the most recent conversation.

As many conversations are shown as will fit on the screen. If they will not all fit, the number of remaining conversations is indicated at the bottom. These conversations can be displayed by means of the DAL command CONTINUE.

Show all active and inactive conversations, beginning with the specified conversation number

Input format: DISPLAY SQL,OPTION=ALL,VG=*con-no*

```
Header line:            VG-NR    TYPE    VG-START          MEM  TA-ID    IDLT
Information line(s):    con-no   type    date              mem  ta-id    time
End line:  {END OF DISPLAY | CONTINUE FOR n FURTHER VGS}
```

Explanations:

con-no Conversation number; up to eight digits;

number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each session segment.

type Type of application which initiated the operation:

UTM
A UDS/SQL-openUTM application

BATCH
A UDS/SQL-TIAM application

date Time stamp for the beginning of the conversation in UDS/SQL:
yyyy-mm-dd hh:mm:ss

mem Amount of SQL-specific memory, in Kbyte; up to four digits

ta-id Global transaction ID of the currently open TA in the conversation; up to eight characters, or "- - -" if no TA is open or has as yet been opened due to the DAL command STOP NEW.

time Duration of inactivity, in minutes; up to four digits, or "- - -" if a TA is open, i.e. the conversation is active

n Number of conversations not yet shown; up to four digits

If the amount of space for a particular output field is insufficient to display the current value, the string ">>...>" is displayed instead.
All conversations are shown, sorted in order of increasing age, beginning with the conversation specified.
As many conversations are shown as will fit on the screen. If they will not all fit, the number of remaining conversations is indicated at the bottom. These conversations can be displayed by means of the DAL command CONTINUE.

Show the conversations inactive for the longest amount of time

Input format: DISPLAY SQL,OPTION=IDLE

```
Header line:            VG-NR    TYPE    VG-START          MEM  IDLT
Information line(s):    con-no   type    date              mem  time
End line:  {END OF DISPLAY | CONTINUE FOR n FURTHER VGS}
```

Explanations:

con-no Conversation number; up to eight digits;

number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each session segment.

type Type of application which initiated the operation:

UTM
A UDS/SQL-openUTM application

BATCH
A UDS/SQL-TIAM application

date Time stamp for the beginning of the conversation in UDS/SQL:
yyyy-mm-dd hh:*mm*:*ss*

mem Amount of SQL-specific memory, in Kbytes; up to four digits

time Duration of inactivity, in minutes; up to four digits, or "- - -" if a TA is open, i.e. the conversation is active

n Number of inactive conversations not yet shown; up to four digits

If the amount of space for a particular output field is insufficient to display the current value, the string ">>...>" is displayed instead.

The conversations inactive for the longest amount of time are shown, sorted in order of increasing duration of inactivity.

As many conversations are shown as will fit on the screen. If they will not all fit, the number of remaining inactive conversations is indicated at the bottom. These conversations can be displayed by means of the DAL command CONTINUE.

Show all inactive conversations that have been inactive for a particular amount of time

Input format: DISPLAY SQL,OPTION=IDLE,TIME=*t*

```
Header line:            VG-NR    TYPE    VG-START          MEM  IDLT
Information line(s):    con-no   type    date              mem  time
End line:  {END OF DISPLAY | CONTINUE FOR n FURTHER VGS}
```

Explanations:

con-no Conversation number; up to eight digits;

number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each session segment.

type Type of application which initiated the operation:

UTM
A UDS/SQL-openUTM application

BATCH
A UDS/SQL-TIAM application

date Time stamp for the beginning of the conversation in UDS/SQL:
yyyy-mm-dd hh:*mm*:*ss*

mem Amount of SQL-specific memory, in Kbytes; up to four digits

time Duration of inactivity, in minutes; up to four digits

n Number of inactive conversations not yet shown; up to four digits

If the amount of space for a particular output field is insufficient to display the current value, the string ">>...>" is displayed instead.

Conversations that have been inactive for at least *t* minutes are shown, sorted in order of increasing duration of inactivity.

As many conversations are shown as will fit on the screen. If they will not all fit, the number of remaining inactive conversations is indicated at the bottom. These conversations can be displayed by means of the DAL command CONTINUE.

4.4.17 Displaying the number of messages to the UDS/SQL DBH (%DML)

```
%DML
```

The %DML command displays the number of messages to the UDS/SQL DBH. It counts not only the DML statements coming from the COBOL DML, CALL DML, SQL or KDBS, but also any requests from the runtime systems or from UDSCON.

4.4.18 Detaching databases, realms and passwords (DROP)

```
DROP {DB=dbname |
      RN=realm-name[,DB=dbname] |

      PW=password |
      ADM=admpassword}
```

dbname

Name of the database

realm-name

Name of the realm

password

password may be up to four bytes in length and is represented as follows:

C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

d:
where *d* is a decimal number (comprising up to eight digits and a sign) that is to be converted to a binary value

The syntax conventions for BS2000 apply (see the commands manuals for "BS2000 OSD/BC", ADD-PASSWORD)

admpassword

May be up to four bytes in length and is represented as follows:

C'*xxxx*':
where *xxxx* comprises one through four alphanumeric and special characters

X'*nnnnnnnn*':
where *nnnnnnnn* comprises one through eight hexadecimal digits

d:
where *d* is a decimal number (comprising up to eight digits and a sign) that is to be converted to a binary value

The syntax conventions for BS2000 apply (see the commands manuals for "BS2000 OSD/BC", ADD-PASSWORD)

The database administrator can detach databases, realms and passwords from a running session by using the DROP command.

Detaching a database

DROP DB=*dbname*

Detaches the designated database. All the realms involved reach a consistency point. This alters the current DB configuration.

If the given database is not attached and no request to attach it is pending, this command is immediately rejected.

If a request to attach a database (original or copy) which is currently not attached exists, this request is cancelled by DROP DB.

When the database is dropped, the number of DML statements per database and the number of input and output operations per database are output to SYSOUT.

Examples

1. DROP DB for the SHIPPING database

```
//EXECUTE-DAL-CMD CMD=DROP DB=SHIPPING
%  UDS0347 UDS ADMINISTRATION: LOGON    = UAD@0E82 (OPDM239,09:30:37/4TTP)
%  UDS0220 UDS RECEIVED COMMAND: DROP DB=SHIPPING (OPOX073,09:30:37/4TTP)
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:30:37/4TTP)
//EXECUTE-DAL-CMD CMD=PERFORM
%  UDS0220 UDS RECEIVED COMMAND: PERFORM (OPOX073,09:30:37/4TTP)
%  UDS0206 UDS ACCEPTED COMMAND  (OPCC012,09:30:37/4TTP)
%  UDS0356 UDS EXECUTION OF ORDERS FOR CONFEXMP STARTED (OPCC309,09:30:38/4TTP)
%  UDS0722 UDS ORDER DROP DB=SHIPPING IN EXECUTION (OPCC283,09:30:38/4TTP)
%  UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE  (OPCC758,09:30:38
/4TTP)
4TTP: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
4TTP: ------------------------------------------------------------------------
4TTP: SHIPPING                   0         67         57          5          5
%  UDS0356 UDS EXECUTION OF ORDERS FOR CONFEXMP TERMINATED (OPCC309,09:30:40/4TTP)
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:30:41/4TTP)
```

2. DROP DB for several databases

```
//EXECUTE-DAL-CMD CMD=DROP DB=SHIPPING
%  UDS0220 UDS RECEIVED COMMAND: DROP DB=SHIPPING (OPOX073,09:30:45/4TTP)
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:30:45/4TTP)
//EXECUTE-DAL-CMD CMD=DROP DB=CUSTOMER
%  UDS0220 UDS RECEIVED COMMAND: DROP DB=CUSTOMER (OPOX073,09:30:45/4TTP)
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:30:45/4TTP)
//EXECUTE-DAL-CMD CMD=PERFORM
%  UDS0220 UDS RECEIVED COMMAND: PERFORM (OPOX073,09:30:45/4TTP)
%  UDS0206 UDS ACCEPTED COMMAND  (OPCC012,09:30:45/4TTP)
%  UDS0356 UDS EXECUTION OF ORDERS FOR CONFEXMP STARTED (OPCC309,09:30:46/4TTP)
%  UDS0722 UDS ORDER DROP DB=SHIPPING IN EXECUTION (OPCC283,09:30:48/4TTP)
%  UDS0722 UDS ORDER DROP DB=CUSTOMER IN EXECUTION (OPCC283,09:30:48/4TTP)
%  UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE  (OPCC758,09:30:48/4TTP)
4TTP: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
4TTP: ------------------------------------------------------------------------
4TTP: SHIPPING                   0         67         50          5          5
4TTP: CUSTOMER                   0         31         22          4          4
%  UDS0356 UDS EXECUTION OF ORDERS FOR CONFEXMP TERMINATED (OPCC309,09:30:48/4TTP)
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:30:49/4TTP)
```

Detaching a realm

DROP RN=*realm-name*
[,DB=*dbname*]

> The DBH notes a request to detach the realm *realm-name* or cancels a pending ADD request. The realm reaches a consistency point before it is detached.
>
> DB=*dbname* need only be specified if the realm name is not unique throughout the DB configuration. It then specifies the intended database.
>
> The DROP RN command cannot be executed if the database has been locked against updating accesses by means of PP DBNAME=*dbname*,SHA. No communication is possible between the DBHs accessing this database in parallel.

Detachment of realms is appropriate in the following cases:

- Device shortages: realms can be detached in a consistent state either logically or physically.
- Hardware errors where the DBH does not automatically detach the realm concerned: following a DROP and subsequent PERFORM command, the database administrator can reconstruct the realm using BMEND while the session is running. In order to do this, all ALOG files required for repairing the realm must be available. If the current ALOG file of the database in question is needed for the repair, a CHECKPOINT command must also be issued.

The DROP DB and DROP RN requests are not executed until the PERFORM command is issued to initiate execution. A failsafe log of the exclusion of a realm is not entered in the DBDIR until the request has been processed in full. A failsafe exclusion is one that remains in effect following session aborts and DBH termination.

The DROP command is the exact counterpart of the ADD command. As long as execution of a pending ADD request has been not yet initiated by means of PERFORM, the ADD command can be canceled by means of DROP.

> The DBDIR realm cannot be detached.

Excluding a file password

DROP PW=*password*

Excludes the given file password from the set of passwords.

When DROP PW is specified, the DBH checks whether the specified file password is actually known. If it is, the DBH cancels the specified file password. If not, the DROP PW command is rejected by the DBH.

File passwords should not be excluded until the corresponding files have been detached (see section “Assigning passwords to UDS/SQL files”).

Excluding an database administrator password

DROP ADM=*password*

Excludes the given database administrator password.

When DROP ADM is specified, the DBH checks whether the specified database administrator password is actually known. If it is, the DBH cancels the specified database administrator password. If not, the DROP ADM command is rejected by the DBH.

The DROP PW and DROP ADM commands are executed immediately without a PERFORM being issued.

4.4.19 Deleting entries from the distribution table (&DROP DISTRIBUTION)

For UDS-D

```
 &DROP DISTRIBUTION,
     {ALL |
      NODE=processor-name,CONF=confname[,DB=dbname][,ALL]
|
      DB=dbname,SS=subschema-name}
```

processor-name

Name of a host;
must be no more than eight characters in length.

confname

The first eight characters of *configuration-name*; must be unique in its first seven characters. The eighth character must not be '@'. Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

dbname

Name of a database;
may be up to 17 characters in length;
must be unique network-wide.

subschema-name

Name of a subschema;
must be no more than 30 characters in length;
must be unique network-wide in its first six characters.

This command deletes entries from the distribution table (see section "Structure of the distribution table").

&DROP DISTRIBUTION,ALL

These operands are used to delete all the entries. When the command has been executed the distribution table is empty. To create a new distribution table, a new input file must be read in using the command &ADD DISTRIBUTION,FILE=...

&DROP DISTRIBUTION,
NODE=...,CONF=...[,ALL]

These operands are used to delete the specified NODE/CONF entry.
The suffix ALL causes all DB and SS entries chained to the NODE/CONF entry to also be deleted.

&DROP DISTRIBUTION,
NODE=...,CONF=...,DB=...[,ALL]

These operands are used to delete the specified DB entry if it is chained to the specified NODE/CONF entry or it is not chained to any NODE/CONF entry.

The suffix ALL causes all SS entries chained to the DB entry to be deleted as well.

&DROP DISTRIBUTION,
DB=...,SS=...

These operands are used to delete the specified SS entry if it is chained to the specified DB entry or it is not chained to any DB entry.

Example

The following assignments are present at each configuration and are reflected in the distribution tables:

PROC1	CONF1	DB1
PROC2	CONF2	DB2
PROC3	CONF3	DB3

The DB administrator enters the DAL command below at configuration CONF2:

```
&DROP DISTRIBUTION,NODE=PROC1,CONF=CONF1,DB=DB1
```

This deletes the DB entry for the database DB1 in the distribution table of configuration CONF2 and has the following effect:

- Application programs that were started with `SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONF2` can no longer access the subschemas of the database DB1.
- Application programs that were started with another configuration are not affected by the change in this distribution table.

> ¡ Changes to the distribution table do not take effect until the next READY statement is issued.

4.4.20 Generating a memory dump (DUMP)

```
DUMP [{ALL | STD | transaction-id}]
```

Default value:

ALL

Depending on the operand specified, the DUMP command does the following:

ALL outputs the entire address space occupied by the DBH to file; when this is completed, processing resumes as normal.

STD A reduced dump is generated, depending on the system status and BS2000 version used. If a reduced dump is not appropriate from the point of view of UDS/SQL, a full user dump is generated. Once the dump is completed, processing resumes as
normal.

transaction-id

This specification is only still supported for compatibility reasons and has the same effect as STD.

The file for a user dump is created in the following format:
[$*userid*.]DUMP.*jobname.tsn.nnnnn*
With a reduced dump (area dump), the file has the following format:
[$*userid*.]SYS.ADUMP.*jobname.tsn.nnnnn*

where

userid:
Current user ID

jobname:
Name of the job

tsn:
Task sequence number

nnnnn:
Five-digit sequence number with leading zeros

¡ While a dump is being written, UDS/SQL ignores any DAL commands other than %TERM.

4.4.21 Generating a memory dump for the linked-in DBH (%DUMP)

For linked-in DBH only!

```
%DUMP
```

Outputs all the memory space occupied by the DBH to a file.

The program is not aborted.

4.4.22 Executing an online DBTT extension (EXTEND DBTT)

```
EXTEND DBTT,DB=dbname, RECR=recordref[,EXT=extnmbr]
```

> You must enter precisely one space between EXTEND and DBTT. No additional spaces are permitted in this DAL command.

dbname

Name of the database comprising the record type for which online DBTT extension is to be performed. The database and the corresponding realm must already be attached at the time the DAL command is issued and online realm extension must be activated.

recordref

Number of the user record type for which online DBTT extension is to be performed.

You can display the *recordrefs* of the record types using the DAL command DISPLAY DBTT-INCR or take them from the BPSIA log.

recordref=1 is used internally by UDS/SQL and cannot be specified here.

extnmbr

Minimum number of entries by which the DBTT is to be extended. This specification must be greater than or equal to 1 and less than or equal to 99999999. If the parameter is not specified, the extension size is calculated internally.

The DAL command EXTEND DBTT performs precisely one online DBTT extension when a record of the corresponding record type is stored whether or not online DBTT extension is activated. However, for this to be possible online extensibility must be active in the realm in question and the realm must be attached. If extension is successful, message UDS0741 is issued.

Since the DAL command is not executed until the next time a record of this record type is stored, the time between the issuing of the command and the execution of the extension depends solely on when new records are stored by the active applications.

You can use DISPLAY DBTT to find out whether there is still a noted EXTEND DBTT request.

If the realm containing the DBTT is detached, then any as yet unexecuted EXTEND-DBTT request is canceled without notification to the user. In particular, such cancellations are performed when the corresponding database is disconnected or at the end of a DBH session.

The EXTEND-DBTT request is executed the next time a record is stored even if this means that two extensions may be performed close together when online DBTT extension is active because a DBTT extension is performed in parallel in another transaction as a result of the activated online DBTT extensibility.

When multiple EXTEND-DBTT request are issued one after the other before the next record of the record type concerned is stored, the DBTT extension is performed once only, namely with the last extension size specified.

As a rule the actual number of DBTT entries by which the database is extended is somewhat greater than the desired number. The actual number depends on the following factors:

- Page size of the database (2K, 4K or 8K)
- DBTT line length of the record type concerned

Depending on these factors there is an upper limit for the possible extension number to which the higher specifications are automatically reduced.
If bottlenecks occur in the DBTT extension, the number of entries by which the DBTT is extended can also be less than the desired number. In all cases the number of DBTT entries by which the DBTT was actually extended will be displayed with a corresponding message.

If a pending EXTEND-DBTT request fails because online realm extension has been suspended or because the maximum size of the DBTT has been reached, then the online extension to the DBTT in question is suspended in the current session segment and a corresponding error message is output. This message is output for each new EXTEND-DBTT request irrespectively of whether or not a similar message has already been issued in the current suspension segment.

If an EXTEND-DBTT request is present when online extension for the corresponding realm is deactivated, then the EXTEND-DBTT request is cancelled without any notification to the user.

4.4.23 Executing an online realm extension (EXTEND REALM)

```
EXTEND REALM,DB=dbname,RR=realmref,EXT=no-pages
```

dbname

Name of the database of the realm for which realm extension is to be performed.

realmref

Number of the realm for which extension is to be performed. All realm numbers are permissible except those for the DBCOM.

no-pages

Number of database pages by which the realm is to be extended.
no-pages=0, 64 ... 16777215
Please note that up to 64 PAM pages may possibly be requested in addition if one or more new FPA extents are needed to manage the newly added pages.
The special specification *no-pages*=0 enables a previous EXTEND REALM request to be revoked, but only if execution has not already begun. As no further EXTEND REALM is outstanding after such a request cancelation, no further message is displayed.

The EXTEND REALM command allows you to perform a one-off realm extension by the specified number of pages during the next free page search irrespective of whether no more free pages are available or only fewer free pages than the limit value specified using ACT INCR. This command does not require a PERFORM statement.

For the EXTEND REALM command to execute successfully it is necessary for the administrator to have assigned the realm adequate storage space or a secondary allocation greater than 0.

The command can also be executed if ONLINE realm extension is not activated with ACT INCR.

After the extension has been performed (whether successfully or not), the unconditional realm extension is concluded, but any ONLINE realm extension activated with ACT INCR is still retained.

When an EXTEND REALM command is issued, the previous EXTEND REALM command is replaced by the newer one provided execution of the previous EXTEND REALM command has not yet begun, i.e. the size specified most recently is used for the extension.

Irrespective of *no-pages*, a realm extension takes place at most up to the maximum possible number of database pages in the realm.

Execution of the EXTEND REALM command is acknowledged with a corresponding message.

The EXTEND REALM command is rejected immediately in the following cases:

- The *dbname* database is not attached.
- The realm number *realmref* does not exist.
- The realm number of the DBCOM was specified as the realm number.
- The specified realm is not attached.

4.4.24 Releasing SQL resources (FORGET SQL)

```
FORGET SQL,VG=vg-nr
```

con-no

Conversation number; up to eight digits;
number for the SQL conversation, assigned internally by UDS/SQL. Conversations are numbered consecutively during each session segment.

The conversation numbered *con-no* is deleted from the UDS/SQL tables. The command is performed when there are no more open transactions in the specified conversation, i.e. when the conversation is inactive. If the TA in the specified conversation is active, the DAL command for the conversation is noted and executed at the end of the TA.

If the conversation is to be deleted despite having an open transaction, an ABORT command must first be issued for the TA.
If the conversation contains a TA that has not yet been opened due to the DAL command STOP NEW, it cannot be deleted until the DAL command GO has been executed.

See also chapter “The SQL conversation”.

4.4.25 Canceling the STOP state of transactions (GO)

```
GO {transaction-id | ALL | OLD}
```

The GO command is the exact counterpart of the STOP command, i.e. depending on the operand specified, it continues the execution of one or more transactions currently in the STOP state:

transaction-id

Identifier for a current transaction;
Cancels the STOP state for the specified transaction.

ALL Cancels the STOP state for all transactions, i.e. the processing of known transactions is resumed, and new transactions are admitted and started.

OLD Cancels the STOP state only for those transactions for which the application program has already issued a READY.

4.4.26 Locking entries in the distribution table (&LOCK DISTRIBUTION)

For UDS-D

```
&LOCK DISTRIBUTION,
   {NODE=processor-name |

    CONF=confname |
    DB=dbname |
    SS=subschema-name }
```

processor-name

Name of a host;
must be no more than eight characters in length.

confname

The first eight characters of *configuration-name*;
must be unique in its first seven characters. The eighth character must not be '@'. Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

dbname

Name of a database;
may be up to 17 characters in length; must be unique network-wide.

subschema-name

Name of a subschema;
must be no more than 30 characters in length; must be unique network-wide in its first six characters.

The following elements can be locked in the local distribution table:

- hosts
  All related NODE/CONF entries are locked.
- configurations
  The related NODE/CONF entry is locked.
- databases
  The related DB entry is locked.
- subschemas
  The related SS entry is locked.

After processing the DAL command &LOCK DISTRIBUTION, UDS-D rejects any READY statement referencing a locked entry.
The application program is assigned the status code 141.

&LOCK DISTRIBUTION blocks access to the specified entry.
For example:
If a subschema is locked, any READY statement referencing that subschema is rejected.
If a configuration is locked, READY statements referencing any of that configuration's subschemas are rejected.
Under local administration &LOCK DISTRIBUTION only changes the local distribution table.

> Changes to the distribution table do not take effect until the next READY statement is issued.

Example

The following assignments are present at each configuration and are reflected in the distribution tables:

PROC1	CONF1	DB1
PROC2	CONF2	DB2
PROC3	CONF3	DB3

The DB administrator enters the DAL command below at configuration CONF2:

```
&LOCK DISTRIBUTION,DB=DB1
```

This locks the database DB1 in the distribution table of configuration CONF2 and has the following effect:

- Application programs that were started with `SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONF2` can no longer access the subschemas of the database DB1.
- Application programs that were started with another configuration are not affected by the change in this distribution table.

4.4.27 Changing the settings of DEFAULT-SUPPORT and RESERVE-SUPPORT of the ALOG files (MODIFY ALOG)

```
MODIFY {ALOG | ALOG-RES},DB=dbname[,VALUE={:catid: | PUBLIC}]
       [,SHARE={YES | NO}]
```

ALOG Modify the settings for ALOG DEFAULT-SUPPORT.

ALOG-RES

Modify the settings for ALOG RESERVE-SUPPORT.

dbname Name of a database (maximum length 17)

:*catid*: BS2000 catalog ID (see section “Using pubsets in UDS/SQL”)

PUBLIC

The ALOG file is created on public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID

SHARE=YES

All users may have access to the file.

SHARE=NO

Only the owner may have access to the file.

> The SHARE operand can be specified for both commands MODIFY ALOG and MODIFY ALOG-RES. However, in the output of the command DISPLAY ALOG the value is only displayed for the default support.

Example

See example for MODIFY ALOG-SIZE.

4.4.28 Changing the storage allocation of the ALOG files (MODIFY ALOG-SIZE)

```
MODIFY ALOG-SIZE,DB=dbname,VALUE=([primary][,[secondary]]
```

dbname

Name of a database (maximum length 17)

primary

Primary allocation, in PAM pages
primary = 192..50337645

secondary

Secondary allocation, in PAM pages
secondary = 576..32767

The DAL command MODIFY ALOG-SIZE changes the storage allocation for ALOG files. Both the primary and secondary allocation can be changed. It will take effect for the creation of the next ALOG file.

Example

```
//EXECUTE-DAL-CMD CMD=MODIFY ALOG,DB=CUSTOMER,VALUE=PUB,SHARE=YES
 %  UDS0220 UDS RECEIVED COMMAND: MODIFY ALOG,DB=CUSTOMER,VALUE=PUB,SHARE=YES (OPOX073,09:31:
32/4TT9)
 %  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:31:32/4TT9)
//EXECUTE-DAL-CMD CMD=MODIFY ALOG-SIZE,DB=CUSTOMER,VALUE=(300,600)
 %  UDS0220 UDS RECEIVED COMMAND: MODIFY ALOG-SIZE,DB=CUSTOMER,VALUE=(300,600) (OPOX073,09:
31:32/4TT9)
 %  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:31:32/4TT9)
 //EXECUTE-DAL-CMD CMD=DISPLAY ALOG
 %  UDS0220 UDS RECEIVED COMMAND: DISPLAY ALOG (OPOX073,09:31:32/4TT9)
 4TT9: USERID    DATABASE-NAME       DEF-SUP  RES-SUP  PRIM-ALL  SEC-ALL  SHARE
 4TT9: ------------------------------------------------------------------------
 4TT9: $XXXXXXXX.SHIPPING              IUDS     ABN2       192      576     NO
 4TT9: $XXXXXXXX.CUSTOMER              IUDS     IUDS       300      600    YES
 4TT9:
 4TT9: UDS/SQL V2.9   2 DATABASES IN CONFEXMP
 %  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:31:32/4TT9)
```

4.4.29 Changing the allocation of volumes for the RLOG files (MODIFY LOG)

```
MODIFY {LOG | LOG-2},
       VALUE={:catid: |
              PUBLIC |
              (priv-vsn-1/device-1[,priv-vsn-2 device-2
               [,priv-vsn-3 device-3]])
              (vsn-1[,vsn-2[,vsn-3]])}
```

LOG Changes the volumes for the original RLOG file.

LOG-2

Changes the volumes for the duplicate RLOG file.

:*catid*: BS2000 catalog ID (see section “Using pubsets in UDS/SQL”).

PUBLIC

The RLOG file or duplicate RLOG file is created on public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID.

priv-vsn

The RLOG file or duplicate RLOG file is created on private disk. Up to three volume serial numbers can be specified, no two of which may be the same. A volume serial number may comprise up to six alphanumeric characters.

device Device type of the private disk. The device type may comprise up to eight alphanumeric characters.

vsn The RLOG file or the duplicate of the RLOG file is created on disks which are allocated to an SF pubset or to a volume set of an SM pubset. Up to three VSNs can be specified, no two of which may be the same. The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). Please also take into account “Additional conditions for VSN specifications”.

You may specify up to three volumes.

A DAL command may be up to 64 characters in length. With long volume specifications the abbreviations MOD and VAL can be used. LOG-2 must be fully written out, to avoid ambiguity.

The DAL commands MODIFY LOG and MODIFY LOG-2 enable you to change the volume allocation for an original or duplicate RLOG file to be newly created. The changes are securely recorded in the session log file (SLF) for use in a session restart.

Whether or not a duplicate RLOG file is to be maintained is specified on starting the DBH. This determination cannot be changed later with the DAL command MODIFY. The DAL command MODIFY can be used to change the volumes for the RLOG files.

When a duplicate RLOG file is maintained the original and duplicate files must be kept on different volumes. The volume specified in the MODIFY command is compared to the existing volume; if the volumes are the same, the command is rejected.

The amount of storage for the RLOG files can be changed with the DAL command MODIFY LOG-SIZE.

Reserve volumes for single and duplicate RLOG files can be specified with the DAL command MODIFY RESERVE.

The new setting must correspond to a pubset which is located within the pubset space of the current UDS/SQL pubset declaration. When a MOD LOG statement is issued between an accepted DAL NEW PUBSETS and the subsequent PERFORM, the new setting must correspond to a pubset which is also located within the pubset space of the new UDS/SQL pubset declaration. If these conditions are not fulfilled, the MODIFY RESERVE command is rejected.

Additional conditions for VSN specifications

The following special rules apply for VSN specifications when the RLOG files are created on disks which are allocated to an SF pubset or to a volume set of an SM pubset:

- VSN specifications are permissible only when authorization exists for physical allocation of public storage space on the pubset concerned.
- In both PUB notation (PUBpxx) and dot notation (pp[pp].[xy]z) it is permissible to omit the sequence number of the disk (xx or [xy]z), in which case only a single VSN specification is possible.

  This enables the RLOG file to be created on any volumes of an SM pubset's volume set.

  Examples:

(PUBX01, PUBX02)	RLOG file on volumes PUBX01 and PUBX02 in the singlecharacter SF pubset X or volume set X of an SM pubset
(PUBX)	RLOG file on any volume in the single-character SF pubset X or volume set X of an SM pubset
(ABC.01,ABC.02)	RLOG file on volumes ABC.01 and ABC.02 in the threecharacter SF pubset ABC or volume set ABC of an SM pubset
(ABC.)	RLOG file on any volume in the three-character SF pubset ABC or volume set ABC of an SM pubset

- If multiple volumes are specified for an RLOG file, these must belong to the same volume set, i.e. the VSNs must contain the same catalog ID. A file cannot extend over multiple volumes sets.
- Twin logging files must be created on different volume sets because when a volume fails the entire volume set is generally no longer available.
- In addition, one of the twin logging files may also not be created on a volume set of an SM pubset if the other is created in this SM pubset on account of a :catid: or PUBLIC specification.
- The following must be borne in mind when the control volume set of an SM pubset is specified for a copy of RLOG or two volume sets of an SM pubset are specified for the RLOG original and duplicate:

  If the control volume set of an SM pubset fails, the entire SM pubset is temporarily no longer accessible. (Reconstruction of the SM pubset is possible.)

4.4.30 Changing the amount of storage for RLOG files (MODIFY LOG-SIZE)

```
MODIFY LOG-SIZE,VALUE=([primary][,[secondary]])
```

primary

Primary allocation, in PAM pages
primary = 1..9999999

secondary

Secondary allocation, in PAM pages
secondary = 1..32767

With the LOG-SIZE command the usual abbreviation rule permitting specification of just the first three characters does not apply, as this would be ambiguous. At least LOG-S must be specified.

The DAL command MODIFY LOG-SIZE is used to change the amount of storage for RLOG files originally specified with the load parameter PP LOG-SIZE. Both the primary and secondary allocation can be changed.

The changes are securely recorded in the session log file (SLF) for use in a session restart.

If the value specified for the secondary allocation is less than the default value 192, the value specified is rounded up to the default value.

4.4.31 Modifying PTCSYNCH values (MODIFY PTCSYNCH)

```
MODIFY PTCSYNCH,
   VALUE=([{WAIT | ABORT | COMMIT}]

          [,[{WAIT | ABORT | COMMIT}]])
```

For UDS-D/openUTM

The **first** value of the PP PTCSYNCH statement applies to secondary subtransactions or openUTM transactions that are in the PTC state at the time of a **warm start**.

For UDS-D

The **second** value of the PP PTCSYNCH statement applies to secondary subtransactions that are in the PTC state in the course of the **current session** and for which the state of the corresponding primary subtransaction cannot be ascertained.

WAIT The secondary subtransactions/openUTM transactions wait until they are terminated either by a message from the corresponding primary subtransaction or openUTM or explicitly by the database administrator using the DAL command ABORT, OPTION=PTC or COMMIT.

ABORT

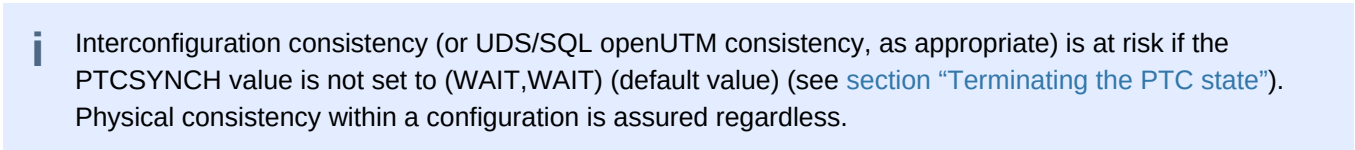
UDS/SQL rolls back the secondary subtransactions/openUTM transactions (FINISH WITH CANCEL) and issues a warning to the database administrator.

COMMIT

UDS/SQL terminates the secondary subtransactions/openUTM transactions, commits the updates (FINISH) and issues a warning to the database administrator.

The DAL command MODIFY PTCSYNCH is used to change the values of the DBH load parameter PP PTCSYNCH dynamically during the session. This makes it possible to terminate secondary subtransactions/openUTM transactions in the PTC condition. If one of the two values is left unspecified, the current value remains unchanged.

It may be necessary to change the default value (WAIT) for warm starts if the database administrator wishes to use a warm start to attach a database in which a secondary subtransaction or a openUTM transaction is in the PTC condition.
By changing the PTCSYNCH value from WAIT to COMMIT or ABORT, the database administrator causes enforced attachment of the database.



Interconfiguration consistency (or UDS/SQL openUTM consistency, as appropriate) is at risk if the PTCSYNCH value is not set to (WAIT,WAIT) (default value) (see section “Terminating the PTC state”). Physical consistency within a configuration is assured regardless.

4.4.32 Changing reserve volumes for the RLOG files (MODIFY RESERVE)

```
MODIFY RESERVE,
       VALUE={NONE |
              :catid: |
              PUBLIC |
              (priv-vsn-1/device-1[,priv-vsn-2 device-2
               [,priv-vsn-3 device-3]]) |
              (vsn-1[,vsn-2[,vsn-3]])}
```

:*catid*: BS2000 catalog ID (see section “Using pubsets in UDS/SQL”).

PUBLIC

Public disk (PUBLIC VOLUME) under the default catalog ID of the configuration user ID.

priv-vsn

The RLOG file is created on the private disks specified as reserve volumes. Up to three volume serial numbers can be specified, no two of which may be the same. A volume serial number may comprise up to six alphanumeric characters.

device Device type of the private disk. The device type may comprise up to eight alphanumeric characters.

vsn Disks which are allocated to an SF pubset or to a volume set of an SM pubset are specified as reserve volumes for the RLOG file. Up to three VSNs can be specified, no two of which may be the same. The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). Please also take into account "Additional conditions for VSN specifications" in chapter "Changing the allocation of volumes for the RLOG files".

You may specify up to three volumes.

A DAL command may be up to 64 characters in length. With long volume specifications the abbreviations MOD, RES and VAL can be used.

The DAL command MODIFY RESERVE is used to change the reserve volumes for files to be newly created, or to specify no reserve volumes. The reserve volumes were originally specified with the load parameter PP RESERVE.

The new volumes are securely recorded in the session log file (SLF) for use in a session restart.

The reserve volumes specified may not be the same as those currently in effect for PP LOG and PP LOG-2. This is checked when a reserve volume is needed.

The new setting must correspond to a pubset which is located within the pubset space of the current UDS/SQL pubset declaration. Between an accepted DAL NEW PUBSETS and the subsequent PERFORM, the new setting must correspond to a pubset which is also located within the pubset space of the new UDS/SQL pubset declaration. If these conditions are not fulfilled, the MODIFY RESERVE command is rejected.

4.4.33 Check and note a new UDS/SQL pubset declaration (NEW PUBSETS)

```
NEW PUBSETS
```

The NEW PUBSETS command is used to check and note a new UDS/SQL pubset declaration. UDS/SQL checks again whether a valid UDS/SQL pubset declaration is available in the job variable with the LINK name UDSPS01. The check takes place immediately. However, the checked UDS/SQL pubset declaration only becomes effective with the next PERFORM (which will possibly be triggered internally).

If the DBH is running in interactive mode, the new UDS/SQL pubset declaration can be made known by assigning another job variable, however, in practice only a modification of the job variable content should be relevant in the case of a master task which is executing in the background.

Modification of the current UDS/SQL pubset declaration using DAL NEW PUBSETS is rejected if the new UDS/SQL pubset declaration does not cover the pubsets which were specified for LOG, LOG-2, RESERVE of RLOG logging and for DEFAULT-SUPPORT and RESERVE-SUPPORT of ALOG logging of every attached database.

The output after a successful check has the following layout:

```
UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
UDS0746 UDS PUBSET DECLARATION (PENDING) FOLLOWS (...,tsn4)
tsn4: UDS-PUBSET-JV: jv-name
tsn4: PUBSETS: catidgroup_1
tsn4: PUBSETS: catidgroup_2
tsn4: PUBSETS: ...
Once each for every catid group specified in the UDS/SQL pubset declaration.
tsn4: DEFAULT PUBSET: <default catalog ID of the execution user ID>
tsn4: ------------------------------------------------------
UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND. (...,tsn4)
```

If no UDS/SQL pubset declaration is assigned, the catid group "*" is used. The following is output to distinguish this from an explicitly specified catid group "*":

```
UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
UDS0747 UDS PUBSET DECLARATION NOT PRESENT, DEFAULT IS CATID * (PENDING) (...,tsn4)
UDS0218 UUDS COMPLETED EXECUTION OF DAL COMMAND. (...,tsn4).
```

The assignment of a UDS/SQL pubset declaration job variable which only contains blanks is permissible; the Default Public Volume Set of the execution user ID is taken into account.

If an error is detected when the UDS/SQL pubset declaration is checked, the UDS/SQL pubset declaration which was previously valid continues to apply.

A DAL NEW PUBSETS command (also one containing errors) deletes any previous DAL NEW PUBSETS command which has existed since the last PERFORM.

Incorrect assignment of a pubset declaration job variable

The assignment of a non-existent or an inaccessible job variable or a job variable with the length 0 is rejected as follows:

```
UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
UDS0752 UDS USER ERROR: ACCESS TO UDS-PUBSET-JV VIA JV-LINKNAME UDSPS01 FAILED (...,tsn4)
UDS0209 UDS USER ERROR: COMMAND REJECTED ERROR IN UDS PUBSET DECLARATION (...,tsn4)
```

Additional information on the cause of the error may be provided in other messages. Content errors in the UDS /SQL pubset declaration are reported as follows depending on the type of error and the time at which it was detected:

- Syntax errors which were detected during the syntactical analysis (identification of the catid groups):

  ```
  UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
  UDS0748 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX): <reason> (...,tsn4)
  ```

  where <reason> can have the following values:

  - CATID GROUP TOO LONG
  - TOO MANY CATID GROUPS
  - EXCLUDE CATID GROUP NOT ALONE

  ```
  tsn4: UDS-PUBSET-JV: jv-name
  tsn4: UDS-PUBSET-JV-CONTENTS:
  tsn4: Content of the job variable may be distributed over several lines
  tsn4: ------------------------------------------------------------------
  UDS0209 UDS USER ERROR: COMMAND REJECTED
  ERROR IN UDS PUBSET DECLARATION (...,tsn4)
  ```

- Errors detected while the catid groups are being checked using the SDF macro CMDWCC:

  ```
  UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
  UDS0749 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX) CATID GROUP REJECTED BY CMDWCC
  (...,tsn4)
  tsn4: UDS-PUBSET-JV: jv-name
  tsn4: PUBSETS: catid-group
  ```

  For each catid group in which the CMDWCC macro detected a fault and which resulted in failure, up to 10 times.
  If there are more than 10 catid groups with errors, only the first 10 are logged, and the line below is used to show that further catid groups with errors exist:

  ```
  tsn4: FURTHER ERRORS NOT SHOWN
  tsn4: ----------------------------------------------------------
  UDS0209 UDS USER ERROR: COMMAND REJECTED
  ERROR IN UDS PUBSET DECLARATION (...,tsn4)
  ```

- Errors detected while the catid groups are being checked using FSTAT:

```
UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
UDS0749 UDS USER ERROR IN UDS PUBSET DECLARATION (SYNTAX) CATID GROUP REJECTED BY FSTAT
(...,tsn4)
tsn4: UDS-PUBSET-JV: jv-name
tsn4: PUBSETS: catid-group, FSTAT-DMS-RC: xxxx
```

  For each catid group in which FSTAT detected a fault and which resulted in failure. The catid group is output in its actual length; the line format is consequently variable.

```
tsn4: ------------------------------------------------------
UDS0209 UDS USER ERROR: COMMAND REJECTED
ERROR IN UDS PUBSET DECLARATION (...,tsn4)
```

- The new UDS/SQL pubset declaration does not cover the pubsets which were specified for LOG, LOG-2, RESERVE of RLOG logging and for DEFAULT-SUPPORT and RESERVE-SUPPORT for ALOG logging of every attached database:

```
UDS0220 UDS RECEIVED COMMAND: NEW PUBSETS (...,tsn4)
UDS0755 UDS USER ERROR: CATID MISSING IN UDS PUBSET DECLARATION (NEW): <loginfo>: (...,tsn4)
```

  The message may be issued several times.
  <loginfo> specifies which catalog ID is missing for which logging object:

  - :catid: , RLOG
  - :catid: , RLOG-2
  - :catid: , RLOG-RESERVE
  - :catid: , ALOG-DEFAULT ,DB: <dbname>
  - :catid: , ALOG-RESERVE ,DB: <dbname>

```
UDS0209 UDS USER ERROR: COMMAND REJECTED
ERROR IN UDS PUBSET DECLARATION (...,tsn4)
```

4.4.34 Selecting new RLOG files (NEW RLOG)

```
NEW RLOG
```

The NEW RLOG command places a request with the DBH to close the RLOG file that is currently open and to create and open a new RLOG file.
The NEW RLOG command is not executed until the request is activated with a PERFORM command.

NEW RLOG must be issued by the database administrator if the DBH fails to create an RLOG file on the disks allocated with the load parameters PP LOG, PP LOG-2 and PP RESERVE. In such cases UPDATE transactions and distributed transactions are blocked. The block can be removed only by means of a NEW RLOG command.

NEW RLOG permits the database administrator to have the DBH change the RLOG file.

4.4.35 Performing pending requests (PERFORM)

```
PERFORM {NOCANCEL | CANCEL}
```

Default value:

NOCANCEL

NOCANCEL

Any open transactions impeding execution of pending requests are allowed to terminate normally. The pending (noted) requests are then executed.

CANCEL

Any open transactions impeding execution of pending requests are rolled back.

Pending requests are generated as a result of the following DAL commands: ACT INCR, ACT DBTT-INCR, ADD DB, ADD RN, CHECKPOINT, DEACT INCR, DEACT DBTT-INCR, DROP DB, DROP RN, NEW PUBSETS, NEW RLOG.

The DAL command PERFORM causes the DBH to execute all pending requests as soon as possible, allowing for consistency and any open transactions. This means that all open transactions that **impede** consistent execution of the pending requests must first be terminated. Such transactions can be displayed by means of DISPLAY USERS. Any new transactions of this type will be rejected with an accompanying status code.

While a consistency point is being set, all new UPDATE transactions are rejected with the status code 12122 or 12124 (the status code 12124 is listed in the case of PP ORDER-DBSTATUS=SPECIAL).

New SQL transactions can execute as RETRIEVAL transactions. SQL UPDATE statements are rejected (SQL CODE -810).

If PERFORM has been issued and all requests have not yet been fully processed, the following commands are rejected: ACT, ADD, CHECKPOINT, DEACT, DROP, NEW RLOG.

An error-free UDS/SQL pubset declaration which was checked beforehand using DAL NEW PUBSETS becomes effective with the next (DAL or DBH-internal) PERFORM. This is indicated by the following message:

```
UDS0754 UDS SWITCHES TO NEW PUBSET DECLARATION NOW
```

The processing of DAL commands is described in the section "Executing DAL commands (EXECUTE-DAL-CMD)" in chapter "UDSADM statements".

> ¡ When PERFORM NOCANCEL has been specified, it may happen that processing of pending requests is unacceptably delayed by protracted transactions. To prevent lengthy delays it is possible to enter PERFORM CANCEL at a later stage.

Note on UDS-D/openUTM

With UDS-D/openUTM it is also possible for distributed transactions to delay execution of pending requests, for example if distributed transactions are left in the PTC condition. Such transactions are not affected by the PERFORM CANCEL command, so as not to compromise interconfiguration consistency or UDS/SQL openUTM consistency. In such cases the DAL command DISPLAY should be used to display the statuses of all current transactions, and if necessary an ABORT or COMMIT command should be used to intervene.

4.4.36 Assigning and changing a password (&PWD DISTRIBUTION)

For UDS-D

```
&PWD DISTRIBUTION,CONF=konfname,
   {PWN=new-password |
    PWO=old-password|
    PWO=old-password,PWN=new-password}
```

Default value:

No password

confname

Name of the configuration to which the password is to be assigned.

new-password

New password for the configuration *confname*.

old-password

Existing password for the configuration *confname*.

PWN=*new-password*

Specifies a new password. The new password is entered into the distribution table. This is possible only if there is no existing password for the specified configuration.

Maximum length: eight characters

A blank terminates the password. Excess input is truncated. Readable characters should be used, but this is not checked.

PWO=*old-password*

Deletes the existing password. The old password in the distribution table is overwritten with blanks.

PWO=*old-password*,PWN=*new-password*

Changes the existing password. The old password in the distribution table is changed to the new one.

If for *old-password* you specify a password that does not match the existing password, the command is rejected and an error message is output.

The password for UDS-D enables you to restrict access to your DB configuration. Remote configurations can access the data of the configuration only if they know the password. The password is entered in the distribution table (see also section “Configuration-based password protection”).

4.4.37 Reactivating online extensibility for a Realm (REACT INCR)

```
REACT INCR,DB=dbname[,RR=realmref]
```

> Only a single blank should be left between REACT and INCR. No more blanks are permitted in this DAL command.

dbname

Name of the database of the realm or realms for which online extensibility is to be reactivated.
The database must be active when the command is executed.
The database must not be in SHARED-RETRIEVAL mode.

realmref

Number of the realm for which online extensibility is to be reactivated. The realm must be activate when the command is executed. The numbers of the activated realms can be displayed with DISPLAY INCR.

The numbers of all user realms and of the DBDIR are permitted. It is not permissible to specify the number of the DBCOM.

For certain errors, the DBH forces a suspension of the online extensibility for a realm (see "Troubleshooting"). However, online extensibility as a permanent, potential property of the realm remains active.

The command REACT INCR cancels suspension of the online extensibility for the realm *realm-ref* of the database *dbname*. If RR=*realmref* is not specified, the command is executed for all user realms which have just been activated and the DBDIR of the database *dbname*.

REACT INCR is executed immediately. No PERFORM is required.

If online extensibility is not suspended for the specified realm(s), the command is ignored.

If the suspension of online realm extension is itself withdrawn then online DBTT extensions are possible again.

Procedure in error situations

The command REACT INCR is immediately rejected with the message UDS0209, if:

- the database *dbname* has not been activated or has been activated in SHARED-RETRIEVAL mode.
- the specified realm number *realmref* does not exist.
- the realm number of the DBCOM is specified as the realm number *realmref*.
- the specified realm has not been activated.

4.4.38 Canceling pending requests (RESET ORDERS)

```
RESET ORDERS
```

This DAL command is used to cancel pending requests whose execution has been initiated with PERFORM (see the DAL command PERFORM in chapter "Performing pending requests (PERFORM)").
This is possible so long as the DBH has not yet started executing these requests.

Following cancellation of any pending requests, transaction operation can be resumed without restriction.
Also, new requests can again be entered by means of DAL commands.

This function is useful primarily in those cases where after a PERFORM command has been entered it is determined that there are transactions that are delaying execution of the pending requests for a long period of time but that should not be rolled back.

The command does not affect any requests present for the activation or deactivation of realm or DBTT extensibility (ACT INCR, DEACT INCR, ACT DBTT-INCR, DEACT DBTT-INCR) because execution of these requests is not impeded by open transactions.

4.4.39 Saving the distribution table (&SAVE DISTRIBUTION)

For UDS-D

```
&SAVE DISTRIBUTION,FILE=[:catid:][$userid.]file-name
```

:*catid*: BS2000 catalog identifier

userid

User identification to which *file-name* is to be assigned

file-name

Name of a SAM or ISAM file

In this case specifying :*catid*: is permitted (see section “Using pubsets in UDS/SQL”).

This command saves the current status of the distribution table to the given file. If this file already exists its contents are overwritten.

UDS-D edits the contents of the distribution table such that the saved file has the same format as an input file for the distribution table (see section “Structure of the input file for the distribution table”).

The saved file then contains database records and subschema records.

Any *dbname*, *confname*, *processor-name* or *subschema-name* which is locked is prefixed by a minus sign.

UDS-D does not save the following distribution table entries:

- passwords
- SS entries that are not chained to any DB entry
- DB entries that are not chained to any NODE/CONF entry

4.4.40 Starting UDS-D operation (&START DISTRIBUTION)

For UDS-D

```
&START DISTRIBUTION
```

This command is used to start UDS-D operation, i.e. to start the UDS-D task UDSCT, when

- the DBH has been started with the DBH load parameter PP DISTRIBUTION=STANDBY
- the DBH has been started with the DBH load parameter PP DISTRIBUTION=START, but UDS-D operation has since been terminated with the DAL command &CLOSE DISTRIBUTION
- UDS-D operation has been abnormally terminated (see section “Restarting UDS-D mode”).

Once UDS-D operation has been started, distributed transactions are possible.

4.4.41 Stopping the execution of transactions (STOP)

```
STOP {transaction-id | NEW | ALL}
```

The STOP command stops the execution of one or more transactions, depending on the operand specified. It places the designated transactions in the STOP state.

The STOP state of a transaction can be canceled by one of the following events:

- GO command
- CANCEL command for the transaction
- CLOSE command
- PERFORM command

transaction-id

Identifier for a current transaction;
stops the specified transaction

NEW Allows new transactions, but does not start them

ALL Stops all currently known transactions; also admits new ones, but does not start them

A transaction is not stopped until it issues a DML statement. If the DBH is currently processing a DML statement of the transaction, the statement is processed to completion; the transaction is not stopped until it issues the next DML statement.

To prevent global deadlocks, new transactions from UTM applications and in secondary subtransactions are rejected with status codes.

4.4.42 Terminating secondary subtransactions in the PTC state (&SYNCHRONIZE DISTRIBUTION)

For UDS-D

```
&SYNCHRONIZE DISTRIBUTION
```

After an event such as a computer crash or a line failure, the database administrator uses &SYNCHRONIZE DISTRIBUTION to try to terminate all secondary subtransactions which are in the PTC condition, in order to reduce wait times for secondary subtransactions. Waiting periods may arise if the value of the DBH load parameter PP PTCSYNCH for the current session is set to WAIT.

To terminate a secondary subtransaction in the PTC state, the DBH of the primary subtransaction needs to be queried to determine whether the primary subtransaction was terminated with FINISH or with FINISH WITH CANCEL so that the secondary subtransaction can be terminated in the same mode.

If the configuration of the primary subtransaction is not accessible, all secondary subtransactions for which it is not possible to ascertain the condition of the primary subtransaction are immediately handled as defined by the PTCSYNCH value for the current session (see section “Monitoring secondary subtransactions in the PTC state”).

If &SYNCHRONIZE DISTRIBUTION is not specified, UDS-D reacts as defined by the PTCSYNCH value for the current session after a period governed by the PP CHCKTIME parameter.

4.4.43 Aborting the session immediately (%TERM)

```
%TERM
```

The %TERM command is used by the DBH to abort the session by the shortest route (emergency halt), dispensing with the normal termination activities.

UDS/SQL can be optionally made to output a partial or complete memory dump (see PP DUMP in chapter "DBH load parameters").

> %TERM should only be used if the master task stops accepting commands due to an error. It always causes an abnormal termination of the session. All currently inconsistent databases in the configuration require a warm start whenever they are to be attached again.

4.4.44 Removing locks from distribution table entries (&UNLOCK DISTRIBUTION)

For UDS-D

```
&UNLOCK DISTRIBUTION,
  {NODE=processor-name |
   CONF=confname |
   DB=dbname |
   SS=subschema-name }
```

processor-name

Name of a host;
must be no more than eight characters in length.

confname

The first eight characters of *configuration-name*;

must be unique in its first seven characters. The eighth character must not be '@'.
Trailing zeros are nonsignificant characters, i.e. there is no difference between *confname* ABC and *confname* ABC0.

dbname

Name of a database;
may be up to 17 characters in length; must be unique network-wide.

subschema-name

Name of a subschema;
must be no more than 30 characters in length;
must be unique network-wide in its first six characters.

Locks can be removed from the following elements in the distribution table:

- hosts
- configurations
- databases
- subschemas

The locked entries in the local distribution table are those which have been locked with the DAL command &LOCK DISTRIBUTION or which have a minus sign prefix in the input file for the distribution table (see section "Structure of the input file for the distribution table").

&UNLOCK DISTRIBUTION removes blocked access to the specified entry.
Under local administration, &UNLOCK DISTRIBUTION only changes the local distribution table.

Example

When the lock on a subschema is removed, READY statements are again accepted for that subschema. When the lock on a configuration is removed, READY statements are again accepted for subschemas of that configuration unless databases or subschemas have been locked explicitly.

> Changes to the distribution table do not take effect until the next READY statement is issued.

Example

The following assignments are present at each configuration and are reflected in the distribution tables:

PROC1	CONF1	DB1
PROC2	CONF2	DB2
PROC3	CONF3	DB3

The DAL command below is entered by the DB administrator at configuration CONF2:

```
&UNLOCK DISTRIBUTION,DB=DB1
```

This removes the lock on the database DB1 in the distribution table for configuration CONF2 and has the following effect:

- Application programs that were started with `SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONF2` will now be able to access all subschemas of the database DB1, except for those which were locked explicitly.
- Application programs that were started with another configuration are not affected by the change in this distribution table.

5 High availability

Data managed in UDS/SQL is generally mission-critical for the company using it. It should therefore be specially protected against data loss and is subject to very high availability requirements. UDS/SQL offers a wide variety of functions – partly in combination with other BS2000 products – to meet this need.

5.1 Uninterrupted database operation

To avoid interruptions to database operation, a number of the database configuration settings can be modified during live operation:

- You can add or drop databases by means of the DAL commands ADD DB and DROP DB.
- You can similarly add or drop individual realms, using the commands ADD RN and DROP RN. This makes it unnecessary in many cases to drop the entire database when performing necessary repair or structuring work, thus keeping the impact on live operation to a minimum.
- You can modify the assignment of volumes for RLOG files which are to be newly created, using the DAL commands MODIFY LOG, MODIFY LOG-2 and MODIFY LOG-SIZE.
- You can modify spare volumes for RLOG files, using the command MODIFY RESERVE.
- You can modify the configuration parameters for distributed processing, using the DAL commands &ADD, &CHANGE, &DROP, &LOCK, &PWD, &SAVE, &SYN and &UNLOCK.
- You can switch ALOG and RLOG files by means of the commands DAL CHECKPOINT and NEW RLOG.
- If the free space in a realm has been depleted by a substantial number of insertions, or if it has fallen below a defined threshold value, the DBH can automatically extend the realm during live operation (see chapter “Resource extension and reorganization during live operation”).
- You can modify an existing UDS-SQL pubset declaration, note the modification using the DAL NEW PUBSETS command, and then activate it using DAL PERFORM.

If the maximum number of records for a record type is reached when new records are stored then the DBH can automatically extend the DBTT during operation (see chapter “Resource extension and reorganization during live operation”).

5.2 Interaction with the transaction monitor openUTM

When restarting an openUTM application with UDS/SQL, the data available in the databases is automatically synchronized with the data of the application to ensure the consistency and integrity of the data. In particular, consistency and integrity are guaranteed in the event of faults, such as power failure or system crash, which may occur during a transaction.

When terminating an openUTM application with UDS/SQL on account of a fault, the transactions which were active at the time are closed in as coordinated a way as possible, so that the subsequent restart involves the minimum of effort. Occupied resources are immediately released so that they can immediately be used again by other applications.

5.3 Formatting the databases

One of the most important concepts of UDS/SQL is that it distributes data from one database across several files, known as realms. This increases the availability of data managed by UDS/SQL, since the effects of device, write or read errors are localized to subunits in the database.

Utility routines do not always need to use all realms to perform updates. This minimizes the potential restrictions of parallel DBH operation.

Copies of databases can be created on several local or remote servers. Replicates are updated with the ALOG files of the original database so that, if the original data becomes unavailable, backup data can quickly be accessed and processed. A further advantage of this distributed data management is that spatially distributed read access can also be accelerated by the replicas (local storage concept).

5.4 Error tolerance

If an error is detected in an application program, the effect remains local. The transaction affected is reset. In the case of more serious errors, the application program is terminated. This prevents other transactions from being impacted.

If STXIT events in application programs cause these programs to terminate, occupied resources are immediately released so that they can immediately be used again by other applications.

5.5 Distributed processing

It is possible to distribute a logically coherent set of data across several BS2000 servers in a computer network. The benefit of using several database servers is that the load of each server is significantly lower than the load produced when using a single database server. Equally, distributed databases limit the maximum damage which can result from the failure of a single database server.

Database distribution is controlled by tables. These can be modified at any time, thus enabling distribution to be adapted dynamically to react to changing situations.

Distributed processing can be started and terminated separately from the independent DBH session. In particular, several UDS-D sessions can be started sequentially in a single independent DBH session, using the DAL commands &START and &CLOSE.

The administrative handling of transactions in PTC status ("prepare to commit") can be modified dynamically. This guarantees completely autonomous administration ("site autonomy"), even in the event of global faults or of a limited availability of network components (see section “Modifying PTCSYNCH values (MODIFY PTCSYNCH)”).

5.6 Media recovery

A shadow database is a copy of an original database to which all updates are applied after a time delay. If the original database is destroyed, a shadow database can be used to quickly resume full database operation after it has been updated using the most recent after images (see section “Shadow database”).

During live database operation, you can perform so-called online backups using BS2000 operating system functions e. g. with the BS2000 utility routines ARCHIVE or HSMS . This makes it possible to create backups without impairing live operation.
Online backups created in this way are generally inconsistent unless administrative measures are undertaken to ensure that the database is consistent at the start of the backup operation and that no changes are made to it during the entire period of the backup.

When a backup has been made using BS2000, it can be updated by means of the corresponding log files and thereby be transformed into a shadow database. UDS/SQL knows internally which log files are required for this update (see section “Saving a database online”).

5.7 Restart functions

Another of the most important functions of UDS/SQL is to ensure the consistency and integrity of data, especially when faults such as a system failure occur during a transaction process. During a restart, UDS/SQL synchronizes the data available in the databases to bring them up to a consistent state (database warm start).

A database warm start can also be performed separately for individual databases.

If the DBH is restarted without special configuration activities, an aborted independent DBH session is resumed in the same status as it was in before it was aborted. This is possible using the session log file (SLF), in which the DBH stores the DBH load parameters of the current configuration (see section “Session log file (SLF)”).

You can regulate the duration of the warm start using the load parameter WARMSTART. This parameter controls the rate at which the blocks of the log files are written back to the database from the buffer during live operation. WARMSTART=VERY-FAST writes the pages to the database very frequently, which significantly shortens the duration of a warm start.

5.8 Using standard BS2000 procedures to increase data security

UDS/SQL can use all major BS2000 procedures to improve data security and availability.

UDS/SQL works smoothly in conjunction with the TimeFinder function of Symmetrix disk systems, enabling a BCV disk mirror to be established and split without interrupting database operation. Depending on the status of the database when it is split, the BCV creates either a consistent copy or an inconsistent "online" copy. The online copy must be brought up to a consistent status by means of log files.

The BS2000 function "Dual Recording by Volume" (DRV) can be used without any restrictions for UDS/SQL files. Should a system crash make it necessary to equalize the data stored on the DRV disks, this can be performed using UDS/SQL resources, i. e. DBH and utility routines, with greater precision and thus more quickly than with the BS2000 standard procedure (see section “Using data mirroring for UDS/SQL”).

The product HIPLEX AF (Highly Integrated Systems Complex) enables high availability applications to be implemented in an availability and load-sharing cluster of several BS2000 servers. This minimizes downtime by automatically detecting system failures and avoiding system restarts. The global transaction recovery facility and restart functions of UDS/SQL within the cluster ensure that no data is lost and no inconsistencies occur if a restart is performed on another computer.

A database backup can be created using BS2000 facilities, usually with the BS2000 utilities HSMS and ARCHIVE. With both utilities, it is possible to create a so-called "online copy" of a database. This procedure saves the database during live operation. Generally, this online backup copy is inconsistent; it can be made consistent, parallel to normal database operation, by applying the ALOG files in offline mode.

5.9 Protection against failure of local resources

In addition to supporting BS2000 standard procedures, UDS/SQL also offers certain independent functions which ensure data security and availability in the event of failure of individual resources.

To meet high availability demands, the RLOG files in UDS/SQL which contain the before and after images of each modification, and which are used for a warm start of the database after a failure, can also be duplicated. This form of redundant data storage offers an alternative to RAID technology or to DRV. If RLOG files are duplicated, UDS /SQL automatically checks, where possible, whether two independent volumes have been specified for the RLOG files; if not, the RLOG files will not be created.

Reserve volumes can be specified for storing RLOG and ALOG files, and then be used when the original volumes fail. This can greatly reduce downtime resulting from the failure of logging volumes. You can manage reserve volumes with the DBH load parameter PP RESERVE, the DAL command MODIFY RESERVE and the BMEND statement START-LOG. When the availability of currently assigned logging files is impacted, the DBH automatically switches to the assigned reserve volumes.

> **!** CAUTION!
>
> Database and logging files must not be write-buffered or write/read-buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

5.10 Using UDS/SQL utility routines

Utility routines which only read data can access databases which are currently being processed, parallel to the DBH.

The UDS online utility enables some functions to be executed online (i.e. during DBH operation) which would otherwise in some cases be implemented in a similar manner by other utility routines. The following functions which can modify a database or how it is processed are offered in the form of DMLs:

- The online DML RELOCATE relocates records and tables in order to increase the occupation ratio of the data pages. Furthermore, you can relieve the load on a realm by relocating the pages of a distributable list to another realm.
- The online DML FPASCAN offers an option to immediately reuse free space which, for example, is released when records or tables are deleted.
- The online DML PREFRLM changes the setting of the preferred realm for a distributable list. When storing member records of this distributable list, the DBH searches in this realm for free pages at the lowest table level (level 0).

A detailed description of the UDS online utility is provided in the "Creation and Restructuring" manual).

The utility routine ONLINE-PRIVACY, which connects to the independent DBH like a normal application program, can dynamically modify access rights in the current session (see manual "Creation and Restructuring", ONLINE-PRIVACY).

Restrictions to database operation resulting from time spent checking the physical consistency of a database, can be reduced considerably by using the utility routine BCHECK to perform a precautionary physical consistency check on a consistent copy of the database. The final physical consistency database check can then be performed incrementally against the consistent and checked copy of the database. In this way, only data which has been modified since the preliminary check is included in the check, and the runtime of the check is significantly less than it would be with a complete check.

The utility routine UDSMON, integrated in UDS/SQL, enables a range of information to be queried regarding current database operation, including the status of currently executed transactions. This information enables special operations to be performed specifically at times when there is a low volume of DMLs and transactions.

A more extensive reorganisation of the data and modifications to the data structures can generally only be performed offline. In order to keep the runtime of such operations as low as possible, the utility routines use a number of optimizations which would be impossible for parallel DBH operation. Due also to the generally high level of complexity of such modifications, priority is given in this instance to the higher reliability of offline operation, rather than to availability.

6 Resource extension and reorganization during live operation

UDS/SQL allows you to extend the user realms and DBDIR as well as DBTTs, and thus the number of possible records for the record type of a user schema, during live database operation. If necessary, DBTT and realm extension is performed automatically by the Independent DBH or the Linked-in DBH. This functionality, which is referred to below as "online realm extension" or "online DBTT extension", makes it possible to prevent interruptions to database operation. This results in a high level of availability of the data managed in UDS/SQL (see also chapter “High availability”).

Administering online extension includes the following steps:

- Checking and setting system requirements for online extension (see section “System requirements for online extension”).
- Activating and monitoring online realm extension and online DBTT extension (see section “Free place administration during realm extension” and section “Activating online DBTT extension”).

Irrespective of the activation of the online realm extension and online DBTT extension, the updating utility routines automatically extend the realms of the database being processed or the DBTTs of the record types in the DBDIR and DBCOM on demand (see section “Automatic realm extension by means of utility routines”).

Relocation and compression of stored data can also be performed during live operation (see section “Online reorganization with the UDS online utility”)

6.1 System requirements for online extension

To make proper use of the online realm and DBTT extension functions, you must check, and if necessary adapt, the relevant realm properties before activating the functionality.

A prerequisite for online DBTT extension is that online realm extension is active for the realm containing the DBTT.

6.1.1 Allocating storage space

If a realm is to be capable of online extension, it must have enough storage space available to it at the time of the extension. This condition is satisfied if the realm possesses a sufficient number of file pages at the time the extension is performed or if dynamic file extension is possible for the realm.

You use the BS2000 command CREATE-FILE to allocate storage space when creating a realm file (see the manual "Creation and Restructuring", Preparing database creation). You can subsequently use the MODIFY-FILE-ATTRIBUTES command to modify the storage space allocation.

A distinction is made between primary and secondary storage space allocations:

- The primary allocation reserves a fixed number of file pages for a new file. In the case of existing files, the primary allocation assigns a fixed number of additional file pages.
- A secondary allocation with a value greater than 0 enables the dynamic extensibility of the file. You can choose any value for the secondary allocation subject to the specific requirements and constraints. It should be harmonized with the scope of the online realm extension that you specify with the ACT INCR command on activation.

Storage space must be assigned to a realm file offline, for reasons related to DMS. It is therefore advisable to monitor the current secondary assignment values of the realm files, so as to be able to adapt them if necessary when the database or the realm has been switched offline for maintenance reasons.

The file size of a realm as understood within DMS does not provide a direct indication of the number of database pages, since the file can contain more PAM pages than are actually used as database pages. During a realm extension, a file extension always takes place if database pages which should logically be added to the realm by the DBH are not in the file. The scope of the file extension depends on the number of this type of database pages and on the secondary assignment which has been set.

6.1.2 DMS-related aspects of online realm extension

Note also the following DMS-related aspects of a realm extension:

Files can only be extended in complete allocation units. The size of the allocation unit is set by the system administrator with reference to the disk format (e.g. 3, 4, 32 PAM pages).

Each time a file extension is successfully completed, DMS doubles the value for the secondary assignment up to a maximum value. This maximum value is determined by the Class 2 system parameter DMMAXSC (global across the system for all private disks, default for pubsets) and by the following commands:

- ADD-MASTER-CATALOG-ENTRY, operand MAXIMAL-ALLOCATION,
- MODIFY-MASTER-CATALOG-ENTRY, operand MAXIMAL-ALLOCATION,
- MODIFY-PUBSET-SPACE-DEFAULTS, operand MAXIMAL-ALLOCATION for pubsets.

If a realm file extension nearly reaches a limit, e.g. because the storage space quota of the user ID has been exhausted, then in some cases there is no partial assignment. Thus, a secondary assignment value which is too high can prevent a realm from being extended by a given number of new database pages, although this realm would otherwise have been extended successfully with a smaller secondary assignment value.

If a realm file for which online extension is to be used has been copied, the current secondary assignment setting must be checked before activating it in the session. This is recommended either when using a backup copy or before applying ALOG files for a shadow database.

If frequent file extensions are made to a realm, the file may become fragmented into a large number of extents. You should take account of this possibility by performing the necessary checks and undertaking suitable measures. For information on these issues, see the manuals "Introductory Guide to DMS" and "SPACEOPT (BS2000)").

6.1.3 Estimating the extensibility of a realm

UDS/SQL can manage a maximum of 16777214 database pages in a realm. This gives the following limit values for the maximum file sizes:

Size of database page	Maximum file size in PAM pages
2 KB	16777214
4 KB	33554428
8 KB	67108856

Table 17: Limit values for file sizes

You can ascertain the logical realm size in the following ways:

- The DAL command DISPLAY FPA will output the current logical realm size NR PAGES and the number of free database pages NR FREE PAGES which the realm contains.
- The command DISPLAY REALM of the utility routine BSTATUS will display the logical realm size (SPACE USED BY REALM / TOTAL NR OF PAGES) and detailed information on the free space which the realm contains.
- The command DISPLAY PAGE ZERO of the utility routine BPRECORD will display the logical realm size (FPA BASE / TOTAL NR PAGES).

Use the BS2000 command SHOW-FILE-ATTRIBUTES to output information on the physical realm size i. e. the number of PAM pages which have already been reserved, and on the current secondary assignment setting (SECONDARY-ALLOCATION) of the realm file. Both these specifications are also up to date for an open realm file.

6.1.4 Effect of utility routines on the online extensibility of realms

If you process a realm for which you have activated online extensibility with the utility routine BPGSIZE, you must ensure that a secondary assignment greater than 0 is specified.

When creating new realms with the utility routine BALTER, you must also bear in mind that online extensibility of these realms should subsequently be activated.

None of the other UDS/SQL utility routines modify the values you have set with the DAL command ACT INCR.

6.1.5 Database layout version and realm layout version

The database layout version identifies the structure of a database; in the same way, the realm layout version identifies the structure of a realm. With UDS/SQL V2.9, you can only process databases and realms with layout version 004.00 and 005.00. You can use the DISPLAY PAGE ZERO statement in the utility routine BPRECORD to output a realm’s ACT-KEY-0 page. As part of this information, the database layout version is output for the DBDIR and the realm layout version is output for the other realms.

6.1.6 Database page formatting during realm extension

The following information on database page formatting applies both to online realm extensions and realm extensions performed offline with the utility routine BREORG.

When the DBDIR is extended, the additional database pages are always formatted. This also applies to the DBCOM, even though it can only be extended offline.

When a user realm is extended, only the newly added database pages which may be referenced by probable position pointers (PPP) are formatted. All other newly added database pages are formatted individually the first time they are used. The number of database pages to be formatted significantly affects the time taken for a realm extension. The number of database pages to be formatted also significantly affects the size of the ALOG file. One, two or three 4KB blocks (for 2, 4 or 8KB databases) are written to the ALOG file for every page to be formatted, depending on the length of the database page. The number of database pages to be formatted should thus be kept as low as possible.

Probable position pointers (PPP) can potentially reference pages outside the database if a realm size has been reduced by the utility routine BREORG (MODIFY-REALM-SIZE statement). This is because, for reasons of performance, the individual probable position pointers (PPP) are not updated during size reduction. After realm size reductions, therefore, you should carry out BREORG runs with the statement REORGANIZE-POINTERS for all user realms.

PPPs can also be updated via REORGPPP DML in UDS online utility during ongoing operation. When all PPPs of all user realms are updated, formatting of new pages during an online realm extension can be avoided.

If the highest page to be formatted in a user realm (cf. HIGHEST PAGE NR FOR FORMATTING in the BPRECORD output of DISPLAY PAGE ZERO) is not greater than the number of pages in this realm, then no probable position pointers (PPP) point to any pages outside the database. Newly added database pages need not therefore be formatted during a realm extension, thus improving the performance of an online realm extension.

If the highest page to be formatted in a user realm is not yet known, it is sufficient to execute one BREORG run with the REORGANIZE-POINTERS statement for each realm of the database.

You can ascertain the formatting status of a realm with the utility routine BPRECORD (see the manual "Recovery, Information and Reorganization", and the section on BPRECORD).

You can find out whether or not a realm’s extension pages have to be formatted as follows:

- using the utility routine BPRECORD (see the manual "Recovery, Information and Reorganization", BPRECORD).
- with the DAL command DISPLAY INCR (see section "DISPLAY INCR - Show information on online extensibility of all activated realms in the database configuration" in chapter "Listing information on the DB configuration (DISPLAY)"). Information about whether formatting is necessary is output for all attached realms, thus allowing you to assess the effects of activating online realm extension.

6.1.7 Free place administration during realm extension

The following information on free place administration tables applies both to online realm extensions and realm extensions performed offline with the utility routine BREORG.

During realm extension, free place administration tables known as FPA extents can be created in addition to the basic free place administration table (FPA base).

Each FPA extent has a size of 128KB. In 2KB-databases it consists of 64 pages, in 4KB-databases of 32 pages and in 8KB-databases of 16 pages. You can output information on the FPA base and the FPA extents with the utility routine BPRECORD.

The fact that there are FPA extents available in a realm does not, however, indicate anything about the current online extensibility. The FPA extents can be created by running BREORG, without the realm ever having been extendible online. It is also possible that the realm had been extendible online in the past but that this attribute has been revoked since then.

You can remove FPA extents with the utility routine BPGSIZE. For each file, BPGSIZE combines FPA base with any FPA extents which may be available to a new FPA base, without changing the page size (see the manual "Creation and Restructuring", BPGSIZE).

6.2 Online realm extension

Once the system-related conditions have been fulfilled, you can activate online realm extensibility for a realm. You do this using the DAL command ACT INCR (see section "Activating online extensibility for a realm (ACT INCR)" in chapter "Activating online extensibility (ACT)"). You must enter two values at activation time:

- The first value specifies the scope of an online realm extension, i.e. the number of database pages that are to be added to the realm on each occasion.
- The second value specifies the minimum number of free pages; if fewer remain UDS/SQL initiates a precautionary online realm extension.

The request is not executed until the PERFORM command is issued.

Online extensibility remains active for the realm until the DEACT INCR command is issued (see section “Deactivating online extensibility (DEACT)”).

You can use the DAL command DISPLAY INCR or the utility routine BPSIA or BPRECORD, to obtain information about the online extensibility of a realm.

When databases are created, online realm extensibility and also online DBTT extensibility for DBTTs are automatically activated by the utility routines BCREATE and BFORMAT for all realms for which a secondary space assignment > 0 was specified (for details see section “Activating online extensibility when creating databases”).

6.2.1 The online realm extension process

The actual extension of a realm is started automatically by the DBH if space is needed to process a DML command in a realm where the number of free pages has fallen below the minimum requirement. The realm is extended within the user transaction of this DML. Although this increases the response time of the DML, the process does not significantly impact general operation.

Since online realm extension is started as a precaution if there are still a certain number of free database pages available in the realm, transactions requiring free space in the realm can continue to run in parallel with the realm extension operation. These transactions find the required storage space in the original part of the realm. For performance reasons, these transactions do not wait until realm extension is completed but adopt the optimistic strategy that they will always find space.

The time taken for online realm extension depends mainly on the number of pages to be formatted during extension (see section “Database page formatting during realm extension”).

Generally, the realm is logically extended by the number of database pages which were specified when online extensibility was activated, up to the maximum number of database pages possible in the realm (see table 17 in chapter "Estimating the extensibility of a realm"). If one or several FPA extents are needed to manage the newly added pages, up to 64 PAM pages more may be required than was specified by the DAL command ACT INCR. New FPA extents are located contiguously at the beginning of the extension area, so that subsequent new free pages are created without fragmentation.

If the realm could be extended, the following message is output:

```
UDS0737 REALM HAS BEEN EXTENDED: (&00) DATABASE-PAGES
```

6.2.2 Troubleshooting

If the realm could not be extended, due to an error reported by DMS, the following message is output together with the relevant DMS error:

```
UDS0700 UDS RESOURCE ERROR (&00) (&01) (&02)
```

If a DMS error occurs during realm extension or if the realm has already reached its maximum size before extension is attempted, the DBH suspends online extension for the relevant realm. This prevents constant failed attempts in the current part of the session to extend this realm. The DML statement which triggered the unsuccessful realm extension may then still be completed successfully, if there is still sufficient storage space in the realm to process the DML statement. The relevant acknowledgement is contained in the status code as usual.

If online extension for a realm is suspended, the following message is output:

```
UDS0740 ONLINE REALM EXTENSION FOR REALM (&00) SUSPENDED
```

6.2.3 Possible responses after an unsuccessful online realm extension

The method of reactivating online extensibility for a realm depends on what caused the error which resulted in the suspension of online extensibility.

If the relevant realm does not have to be dropped to clear the cause of the error, you can reactivate online extensibility for this realm in the current part of the session with the DAL command REACT INCR. This is the case, for instance, if additional disk storage space had to be made available under the user ID.

If the relevant realm has to be dropped to clear the cause of the error, online extensibility can only be reactivated once the cause of the error has been cleared, by re-attaching the realm. This is the case, for instance, if the secondary assignment of the realm file has to be reset from 0 to a value greater than 0. Online extensibility of a realm is also implicitly reactivated if the database had to be dropped and restarted to clear an error, or if the session had to be terminated and restarted.

With the DAL command DISPLAY INCR, you can query whether online extension has been suspended for a realm.

6.2.4 DAL commands and utility routines for online realm extension

The following table lists the tasks involved in administering online realm extension together with the available utility routines and DAL commands for each of them.

Function/structure	DAL/utility routine	Parameter/statement
Activating online realm extension	DAL	ACT INCR
Performing online realm extension once only when the next free space search takes place in the realm	DAL	EXTEND REALM
Obtaining information on activating online realm extension of the DBDIR or of a user realm	DAL	DISPLAY INCR
	BPSIA	DISPLAY SCHEMA
	BPRECORD	REALM NAME IS *realm-name* DISPLAY PAGE ZERO
Deactivating online realm extension	DAL	DEACT INCR
Determining whether online realm extension is suspended	DAL	DISPLAY INCR
Reactivating online realm extension	DAL or detach realm, eliminate cause of error and reactivate realm	REACT INCR
Determining logical size of a realm	DAL	DISPLAY FPA
	BPRECORD	REALM NAME IS *realm-name* DISPLAY PAGE ZERO
	BSTATUS	DISPLAY REALM *realm-name*
Determining if formatting is carried out during realm extension	DAL	DISPLAY INCR
Determining page up to which formatting is carried out during realm extension	BPRECORD	REALM NAME IS *realm-name* DISPLAY PAGE ZERO
Determining location and number of FPA extents in DBDIR, DBCOM or in a user realm	BPRECORD	REALM NAME IS *realm-name* DISPLAY PAGE ZERO

Combining FPA base and FPA extents	BPGSIZE	
Determining number of a realm (required for ACT INCR and DEACT INCR)	DISPLAY INCR	
	BPRECORD	

Table 18: DAL commands and utility routines for administration of online realm extension)

6.2.5 Activating online extensibility when creating databases

When databases are created, online extensibility for all realms for which a secondary assignment > 0 was specified for storage space allocation is activated automatically by the utility routines BCREATE and BFORMAT.

Activation of online extensibility therefore applies for all newly created realms with a secondary assignment > 0 from the outset and does not have to be set using the relevant DAL commands. However, you can suppress the activation of online extensibility by defining a secondary assignment = 0 for the realm. Changing the secondary allocation after the utility routines BCREATE and BFORMAT have been executed does not automatically result in the activation of the online extensibility.

If online realm extension is activated in this manner, the default values *nr-pages*=64 and *min-pages*=0 are entered for *nr-pages* and *min-pages* (see section “Activating online extensibility for a realm (ACT INCR)” in chapter "Activating online extensibility (ACT)").

Online DBTT extension is also activated during database creation for all DBTTs which are located in realms for which online extensibility was activated in this manner. The default value SCAN=YES is used for the SCAN parameter here (see section “Activating online DBTT extensibility for a record type (ACT DBTT-INCR)” in chapter "Activating online extensibility when creating databases").

Subsequent modification of the settings (in particular deactivation) for both online realm extension and online DBTT extension using the DAL command is possible.

Irrespective of the online extensibility of a realm, the DAL command EXTEND REALM can be used to note a one-off realm extension which will be performed when the next free space search takes place in the realm.

6.3 Online DBTT extension

The following section describes how you activate online DBTT extension, how it operates, what happens if an error occurs and what DAL commands and utility routines are available for online DBTT extension.

6.3.1 Activating online DBTT extension

Before online DBTT extension is possible, online realm extension must be active for the realm in which the DBTT is located

If online realm extension is active and the system-related conditions for realm extension are fulfilled then you can activate online extensibility for a DBTT. You do this using the DAL command ACT DBTT-INCR (see section "Activating online DBTT extensibility for a record type (ACT DBTT-INCR)" in chapter "Activating online extensibility (ACT)").

The request is not executed until the PERFORM command is issued.

The online extensibility of the DBTT remains active until the DEACT DBTT command is issued for this DBTT (see section “Deactivating online extensibility (DEACT)”).

You can use the DAL command DISPLAY DBTT or the utility routine BSTATUS to obtain information on the online extensibility of a DBTT.

When databases are created, online DBTT extension is activated automatically for all DBTTs which are located in realms for which online realm extension is automatically activated by the utility routines BCREATE and BFORMAT. For details see section “Activating online extensibility when creating databases”.

6.3.2 Execution of an online DBTT extension

The actual extension operation for a DBTT is initiated automatically by the DBH if a search for a free entry reveals that the last entry is occupied (SCAN=NO) or that there are no more free entries in the entire DBTT (SCAN=YES). The DBTT extension is performed within this DML’s user transaction. Although this increases the DML’s response time, the impairment to overall operation due to this process is small.

The period required for online DBTT extension is largely dependent on the time required for the associated online realm extension.

In the case of an online DBTT extension, the realm containing the DBTT is extended in all cases. This obligatory realm extension represents only a very small burden for ongoing transaction operation.

The realm is only extended by the necessary amount, not by the values specified on activation of online realm extension. This means that no superfluous empty pages are created as a result of an online realm extension for a DBTT.

If it was possible to extend the DBTT, the following message is output:

```
UDS0741 DBTT HAS BEEN EXTENDED: (&00) DBTT ENTRIES
```

The insert (&00) indicates the record name and the number of DBTT entries by which the DBTT was extended.

If a number of entries was specified in the EXT parameter, then the DBTT will be extended by a rounded up number of DBTT entries required for DBTT extension. The actual number of DBTT entries depends on the page size of the database and the DBTT line length of the record type concerned.

6.3.3 Initiating an explicit online DBTT extension

You can use the DAL command EXTEND DBTT (see "Executing an online DBTT extension (EXTEND DBTT)") to initiate an online DBTT extension the next time a record is stored in the corresponding record type irrespectively of the activation state of online DBTT extension. In this case, the prerequisites for online realm extension must again be fulfilled.

6.3.4 Response in error situations

If the DBTT could not be extended then a message specifying the reason for this is output.

If the failure of the DBTT extension was due to the failure of the necessary realm extension then further extension attempts for this DBTT are ignored until the realm extension suspension has been terminated.

If a failed online realm extension results in a suspension due to a high extension granularity then you may still be able to perform a successful online DBTT extension using the DAL commands `REACT INCR` and `EXTEND DBTT`.

6.3.5 DAL commands and utility routines for online DBTT extension

The following table lists the tasks involved in administering online DBTT extension together with the available utility routines and DAL commands for each of them.

Function/structure	DAL/utility routine	Parameter/statement
Activating online DBTT extension	DAL	ACT DBTT-INCR
Performing online DBTT extension once only when storage next takes place in the record type	DAL	EXTEND DBTT
Obtaining information on activating online DBTT extension	DAL	DISPLAY DBTT-INCR
	BSTATUS	DISPLAY RECORD
Deactivating online DBTT extension	DAL	DEACT DBTT-INCR
Determining whether online DBTT extension is suspended for a record type	DAL	DISPLAY DBTT-INCR
Combine DBTT base and DBTT extents	BPGSIZE	
Determine a record type’s reference (necessary for ACT, EXTEND and DEACT DBTT)	DAL	DISPLAY DBTT
	BPSIA	

Table 19: DAL commands and utility routines for administration of online DBTT extension

6.4 Online reorganization with the UDS online utility

The UDS online utility enables some functions to be executed online (i.e. during DBH operation) which are otherwise in some cases implemented in a similar manner by other utility routines.

You can compress data offline as follows:

- with BPGSIZE (see the "Creation and Restructuring" manual) or
- by unloading and reloading the data using BOUTLOAD/BINILOAD (see the "Creation and Restructuring" manual) or
- by creating new tables with BREORG (see the "Recovery, Information and Reorganization" manual)

The UDS online utility enables you to relocate and compress data records and small tables specifically in individual realms during live operation.

For distributable lists you can relocate level 0 pages to another involved realm during ongoing operation. During ongoing operation you can change the realm in which free space is searched for for additional level 0 pages which are required (preferred realm).

You can also use the UDS online utility to enable the comparable functions SET and RESET of the BMODTT utility routine (see the "Recovery, Information and Reorganization" manual) during live operation.

The UDS online utility also enables you to update PPPs to the most recent values during ongoing operation.

A detailed description of the UDS online utility is provided in the "Recovery, Information and Reorganization" manual.

6.5 Automatic realm extension by means of utility routines

Utility routines with linked-in DBH

The UDS/SQL utility routines DDL compiler, SSL compiler, BPRIVACY, BGSIA and BGSSIA process the DBDIR and DBCOM (and if necessary required COSSD), but not the user realms of the database involved. When these utility routines with linked-in DBH (= utility sessions) are executed, both the DBDIR and the DBCOM are extended dynamically if necessary. This extension is implemented independently of online extensibility being activated for these realms. However, a prerequisite is that a secondary allocation > 0 is specified in the storage space allocation for the DBDIR and/or the DBCOM. You can thus, if required, suppress automatic extension of the DBDIR or DBCOM by specifying a secondary allocation = 0.

Activation of online extensibility of the DBDIR using a DAL command is only effective for user sessions. You can still modify this activation as before using the DAL command ACT INCR or DEACT INCR of the Independent DBH. You can only influence online extensibility of the DBCOM by means of the secondary allocation.

In utility sessions with linked-in DBH extensions are performed using the default value 64 for ADDITIONAL_NR_PAGES and the value 0 for MIN_NR_FREE_PAGES for the following reasons:

- The DBH only ever requests single free pages in the free page search. No large extension steps are therefore required, and neither the DBDIR nor the DBCOM becomes particularly large.
- There is no parallelism in utility routine runs; consequently no allowance must be made in the form of MIN_NR_FREE_PAGES > 0.

The DBTTs of the record types in the DBDIR and DBCOM (with the exception of the DBTT of the SSIA_RECORD) are, if necessary, also automatically extended independently of activation of online extensibility which is only required for user sessions provided the requirement for the corresponding realm extension (secondary allocation > 0) is satisfied.

The dimensioning of the record types in the DBDIR and DBCOM determined at database creation is retained. The DBTTs of these record types are generally sufficiently large when they are created. The aim here is to ensure that extension of the DBTTs of these record types is restricted to exceptional cases. Otherwise the DBDIR and DBCOM would increase disproportionately in size owing to frequent DBTT extensions.

Utility routines without DBH

The UDS/SQL utility routines BCREATE, BCHANGE, BALTER, BRENAME, BREORG, BFORMAT and BINILOAD and BREORG automatically extend the realms of the database being processed when required.

Regardless of whether online extensibility is activated for these realms, the aforementioned utility routines always cause the realm to be expanded when the available free space is no longer sufficient to execute the utility routine.

However, a requirement for this is that a secondary allocation > 0 has been specified in the storage space allocation for the realm concerned. Consequently you can, if you wish, suppress automatic realm extension by specifying a secondary allocation = 0.

The general conditions described in section “DMS-related aspects of online realm extension” also apply.

The realm concerned is extended by as many database pages as fit into the secondary allocation currently set in DVS, but at least by the default value of 64 pages (another 64 pages can be added if a new FPA extent is required).

If the request for a larger contiguous area of free space in the existing realm cannot be satisfied when the utility routine is executed, the size of this area is taken as the yardstick for the extension (here, too, the number of pages which can be fitted into the secondary allocation and the default value 64 are not fallen below).

The execution of automatic realm extension, its scope and the new size of the realm are shown by the following messages:

```
0074 REALM <realmname> HAS BEEN EXTENDED BY nnn DATABASE-PAGES
     NEW NR OF PAGES : mmm
```

Failure of automatic realm extension results in the following message being issued:

```
0073 DYNAMIC EXTENSION BY nnn DATABASE-PAGES NOT POSSIBLE FOR REALM
<realmname>
```

> ¡ Online copies which are created while a realm is being extended automatically by a utility routine are unusable because when it applies the changes BMEND cannot determine correctly that a consistent DB status has been achieved. Online backup capability of the database is consequently temporarily withdrawn by UDS during such utility routine execution. This prevents online backups using HSMS /ARCHIVE in this period. When COPY-FILE is used, the user is responsible for dispensing with online backups while these utility routines execute.

7 Saving and recovering a database in the event of errors

This chapter explains how to guard against data loss and what actions can be taken if errors occur.

- The UDS/SQL recovery concept
- Transaction recovery
- Media recovery
- Selecting a recovery concept
- Examples

7.1 The UDS/SQL recovery concept

One of the most important functions of a database system is to ensure the consistency of the data at all times. This includes both the physical storage consistency and the logical consistency of the data in the database. First and foremost, the system must command suitable measures to prevent the loss and corruption of data. If errors should occur during database operation, swift recovery to a state of consistency must be possible.

All of the measures directed toward maintaining and restoring data consistency form part of an integrated recovery concept. These measures fall into two categories:
transaction recovery and *media recovery*.

Whereas the purpose of transaction recovery is to ensure the consistency of the data while the system is running, media recovery is designed to maintain the consistency of the data for the entire lifetime of the database, or to restore the database to a consistent state if the need should arise.

The following paragraphs briefly describe which type of recovery is used for different types of error.

Errors on external storage

An external storage medium (disk) can no longer be processed.

- In the event of write and read errors, any open transactions are rolled back using transaction recovery. Restricted database operation takes place until the defective device and its data are replaced.
- In the event of hardware errors, you must return to a backup status created using media recovery (and, if necessary, reconstruct an earlier status) before you resume database operation on these data resources.

System errors

The contents of main memory have been lost due to system failure, but external storage still remains intact.

Any open transactions are rolled back using transaction recovery. Normal database operation is then resumed.

Errors in the application program

Depending on the type of error affecting the data resources, either you must return to a backup status created using media recovery (and, if necessary, reconstruct an earlier status) or the open transaction involved is rolled back by means of transaction recovery.

The major features of the UDS/SQL recovery concept are:

- prompt detection of errors
- confinement of errors to the smallest possible subunits
- fast, flexible handling of errors, and the capability to make repairs to the database while the system is running.

7.2 Transaction recovery

Transaction recovery handles the following types of error:

- Errors in the application program

  This includes all situations in which the current transaction of the application program has to be rolled back, e.g. if the application program crashes, a deadlock occurs or the transaction is rolled back by the DML statement `FINISH WITH CANCEL`.

- Errors in the system

  This means all errors that occur in the DBH, e.g. a UDS/SQL system crash, a BS2000 system crash or a shortage of resources. File errors are not included in this category; these errors are handled using media recovery.

A prerequisite for using this type of recovery is that you have saved changes to your database using both the ALOG and the RLOG file.

7.2.1 Error types and their recovery

This chapter contains the following topics:

- Errors in the application program
- Errors in the system

7.2.1.1 Errors in the application program

These errors are recovered by rolling back the open transaction involved. Transactions are rolled back with the aid of before-images (before-images describe the state of data before it is updated). The before-images are initially stored in main memory and are written as necessary to the RLOG file (see section “RLOG file”).

7.2.1.2 Errors in the system

Error recovery must be capable of performing the following steps in the event of an error in the system:

- It must be able to roll back all open transactions. As in the case of an error in the application program, this is done with the aid of before-images.
- The results of all completed transactions that are not yet in the database must be committed to the database. This is done with the aid of after-images (after images describe the state of data after it has been updated). To this end, the after-images of each transaction are written to the RLOG file, at the latest by the end of the transaction.

Warm start

Following a DBH crash, the databases involved are inconsistent if they have been updated during the session. If these databases are attached to a configuration, UDS/SQL first performs a warm start for these databases. This involves rolling forward completed transactions and rolling back open transactions. The databases are then flagged as consistent.

If a openUTM application is involved, the statuses of the individual transactions must be synchronized with openUTM, for which purpose a status file is required (see section “Status file”).

If a UDS-D application is involved, it is also necessary to synchronize global transactions. Global transactions are transactions for which DML statements are processed in more than one DBH. The status file is required in order to perform this synchronization, too.

Following a DBH crash, you may not update the databases involved until a warm start is performed. This rule does not apply in the following cases:

- The database is destroyed during or after a DBH crash. You must then read in a backup and use the BMEND utility routine to update the database to the time of the crash (UPDATE-DATABASE, DEADLINE = BREAK-POINT statement).
- After the crash, only DMS catalog entries for the files involved, but not their contents, are updated (e.g. following file reorganization by the computer center and swapping out and in by HSMS).
  In this case, the DBH may reject the warm start. If this happens, you must use the BMEND utility routine to update the crashed database to the time of the crash (statement UPDATE-DATABASE, DEADLINE = BREAK-POINT).

Session restart

UDS/SQL stores information on the current configuration in the SLF (see section “Session log file (SLF)”). If the DBH is started after a crash with the same configuration name, the DBH takes the DBH load parameters from the SLF. The DBH load parameters that you specified when starting the DBH are then ignored.
This procedure is called session restart.
If you do not want a session restart, you must delete the SLF before starting the DBH.

7.2.2 Handling the required files

This chapter contains the following topics:

- RLOG file
- Session log file (SLF)
- Status file

7.2.2.1 RLOG file

The RLOG file is required for

- rolling back open transactions during operation or for a warm start
- rolling forward completed transactions for a warm start

The RLOG file therefore contains the before-images and after-images for each update. The RLOG file is created by the DBH at the start of a session. It consists of two physical files with the names

confname.RLOG.*time-stamp*.1
confname.RLOG.*time-stamp*.2

You can use DBH load parameters or DAL commands to determine the disks on which you want the RLOG file to be created. For recovery reasons it is advisable to maintain duplicate RLOG files. You can either use DRV for this purpose or have UDS/SQL maintain a duplicate RLOG file. In this case, the DBH creates two more files with the following names:

confname.RLOG.*time-stamp*.1.SAVE
confname.RLOG.*time-stamp*.2.SAVE

The DBH deletes the RLOG file once the associated session has been terminated without errors. Following a crash, the associated RLOG file is not deleted until a warm start has been performed for all databases affected by the crash.

The DBH protects the RLOG file with an internal password.

Selecting the volume for the RLOG file

When selecting the volumes for the RLOG file, it is important to take recovery and performance aspects into account.

Security

For security reasons the RLOG file should not be placed on the same disks as database files.

When the RLOG file is maintained in duplicate by UDS/SQL, a check is made to see whether you have specified two independent volumes as the storage location. If you have not, configuration of the RLOG files is rejected.

In the case of RLOG files which have been created in duplicate on SF pubsets which are buffered in the global storage, a check is also made to see whether they are buffered in different units. If they are not, configuration of the RLOG files is rejected.

The check cannot be performed by UDS/SQL for RLOG files on SM pubsets because the relevant information cannot be ascertained for SM pubsets. UDS/SQL therefore can no longer completely prevent the existence of "single points of failure".

As the RLOG files are extremely important for ensuring the consistency of the data during ongoing operation, each RLOG file is protected by a password of its own. Generally the DBH deletes RLOG files when they are no longer required. If in exceptional circumstances an RLOG file has to be deleted for other reasons, you can issue the password using the RLOGPASS utility routine in the UDS-SQL-T delivery unit. This password is designed not only to protect the contents of the files against unauthorized access; it is also used to prevent the files from being inadvertently deleted.

Performance

If the RLOG file causes a bottleneck in an application, you can take the following measures:

- Create the RLOG file on a disk that is not otherwise accessed. You should create the RLOG file in such a manner that writing can take place as quickly as possible and without being influenced by other activities on the system.
- If you require an even higher throughput rate, you can store the file on a faster device (e.g. SSD, see the manual "Introduction to System Administration"). Another option is to store the RLOG file on a public disk that is buffered via the global storage (GS).

> **!** CAUTION!
>
> The RLOG files must not be write-buffered or write/read-buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

7.2.2.2 Session log file (SLF)

The DBH stores the DBH load parameters of the current configuration in the SLF, which it creates at the start of a session.

In the event of a session restart, the DBH takes the load parameters of the crashed session from the SLF, ignoring the DBH load parameters that you specified at the DBH start. If you do not want a session restart, you must delete the SLF beforehand. This file is protected with the default password C'UDS'BLANK''. This password is not designed to protect the contents of the file which is employed in various sessions; it is used to prevent them from being inadvertently deleted.

> ! CAUTION!
>
> The SLF must not be write-buffered or write/read-buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

7.2.2.3 Status file

The status file contains information required for, among other things, the synchronization with openUTM and the synchronization of global transactions by UDS-D. The status file is continued in successive sessions. It can be deleted if all openUTM applications active in a configuration or connected via UDS-D have been terminated normally and no more warm starts must be performed for these applications.

The status file is always maintained in duplicate. The following file names are used:

confname.DBSTAT
confname.DBSTAT.SAVE

You must create the status file. DBH protects these files with the default password C'UDS'BLANK''.

If you want to delete the status files, you must specify the OPTION=*DESTROY-ALL operand in the DELETE-FILE command.

> **!** CAUTION!
>
> The status files must not be write-buffered or write/read-buffered in volatile media (see section “Using DAB caching for UDS/SQL”).

7.3 Media recovery

Media recovery includes all the measures which serve to safeguard the data for the entire life of the database. These measures include a preventive backup of the database, the maintenance of ALOG files, and database reconstruction.

A preventive backup involves saving all the data of a database at regular intervals. If, for example, a database disk fails, you can then return to this older backup status without losing all your data.

To avoid losing any updates made to the data between each backup, ALOG (archive log) files are maintained during operation. These files contain the data after it has been updated (after-images). If updates are made to the database but no ALOG file is maintained, a "logging gap" exists for this period of time.

ALOG files allow you to reconstruct database consistency, e.g. for the period between two backups for the entire database. This is done by using the UDS/SQL utility routine BMEND (UPDATE-DATABASE statement) to apply updates in the ALOG file(s) to a backup status of the database until the desired consistent status is achieved.

7.3.1 Making a database backup

To make a backup of a database, i.e. of all UDS/SQL database files, you do not need a database-specific archiving program. You can make a backup of your database using the options available in BS2000, usually the BS2000 utility routines HSMS or ARCHIVE (see the HSMS manual "Volume 1: Functions, Management and Installation" and "ARCHIVE (BS2000)").

Your own organizational requirements determine how often you save the entire database and how you archive it. In addition to making a periodical backup, it is advisable to make a backup when, for example, a logging gap has occurred or the database has been restructured.

7.3.1.1 Consistency points

The data of a database is consistent at certain points in time (known as *consistency points*), i.e. all updating transactions are complete and the updates made by them have been committed to the database.

A consistency point is reached

- on the normal end of a DBH session or a modifying utility routine
- following a successful session restart
- when the ALOG file is switched (e.g. due to overflow)
- with the DAL command CHECKPOINT (only possible with independent DBH)
- when the DBH handles certain events (e.g. deactivates a realm)

The ALOG file may be switched at the same time as a consistency point is reached. The consistency point (checkpoint) is then written. You can update the backup of a database up to this checkpoint using the BMEND utility routine and the completed ALOG file.

Leaving the consistency point when updating a database also signifies the start of the log interval for an empty ALOG file. The end of the log interval is set on reaching the consistency point. If the ALOG file is not switched now and updates continue to be written up to the next consistency point, interval of the ALOG file is increased by this time period. The end of the log interval is updated accordingly.

7.3.1.2 Saving a consistent database

You can save the entire database (without the ALOG files) if the database is in a consistent state (i.e. it has reached a consistency point) and no database operation is taking place on it, i.e. it is not being processed. If, in the event of errors, you have to access this backup, you must first read it in, e.g. using HSMS or ARCHIVE again, before you can resume database operation based on this backup status.

When you save the database, all the options offered by programs such as HSMS or ARCHIVE (for saving to disk, tape or magnetic tape cartridge) are available to you.

7.3.1.3 Saving a database online

HSMS (operands SAVE-OPTIONS=*PARAMETERS(SAVE-ONLINE-FILES=YES)) or ARCHIVE (operands PARAMOLS=YES) can be used to save the UDS/SQL database files while the database is being processed and updated by DBH.

Prerequisites for an online database backup

- You must have declared that the database can be saved online using the utility routine BMEND (ENABLE-ONLINE-COPY statement) before starting database operation.
- AFIM logging (After Image Logging) must be activated for the database since a consistent database state is reflected only by the online backup in conjunction with the ALOG file(s) generated during the save operation. The online backup itself is usually inconsistent unless administrative measures are taken to ensure that the database is consistent at the start of the backup operation and that no changes are made during the entire period of the backup.
  To establish a consistent state on the basis of an online backup, you should first restore the backup to disk before using the utility routine BMEND to update it using the associated ALOG file(s) (UPDATE-DATABASE statement).

Creating online backups

To create online backups of databases or individual realms, you should always use the corresponding HSMS or ARCHIVE statements.

> **!** CAUTION!
>
> When COPY-FILE is used to generate an online copy, the system does not check whether "online backup capability" is enabled for the database concerned. In this case you must therefore ensure yourself that AFIM logging is enabled so that the online copy can be made consistent later by applying the changes.

When you start a modifying UDS utility routine, you must make sure that any online backup operations started beforehand for the database to be processed are terminated, otherwise the online copy might be unusable.

It is not permissible to generate online copies while modifying utility routines are running.

Online backup capability of a database copy

Whether or not a database can be backed up online is recorded both in the UDS administration data and in the database files' DMS catalog entries. When a database is copied, either with COPY-FILE or with HSMS/ARCHIVE, the specifications concerning the database's online backup capability may be lost depending on the parameters that are used. As a result, before using a database copy of an original for which the online backup capability was active, you must ensure that this property is consistent, e.g. by running the utility routine BMEND again with the ENABLE-ONLINE-COPY statement.

7.3.1.4 Shadow database

A shadow database is a consistent copy of the database in which all database file names have the suffix *copy-name*.

The shadow database can be concurrently used with the original database being processed as follows:

- It can be read, e.g. to generate information or statistics, without hindering database operation.
- It can be updated continually with the after-images of the completed ALOG file(s) of the original database. This provides you with relatively current data resources in the event of an error.

You can create a shadow database by copying the files of the original database (without the ALOG files) or by renaming another backup that was read. The file names of the database files of a shadow database are given a freely selectable suffix *copy-name*. The following general rules apply to the shadow database:

db-file-name.*copy-name*

db-file-name

Name of the database file (e.g. *dbname*.*realm-name*)
The specification :*catid*:$*userid*.*dbname*.*realm-name*.*copy-name* may be no more than 54 characters long.

copy-name

Suffix ("name") of the shadow database. *copy-name* may be no more than seven characters long.

If the online backup capability was set for the original database, you should note the comments set out in section “Online backup capability of a database copy” in chapter "Saving a database online".

The shadow database has **no** separate ALOG files. It is updated using the ALOG files of the original database.

> Note that changes to the DBDIR for shadow databases are not recorded in the ALOG file.
> If you modify the DBDIR of the shadow database by administrative measures (e.g. with the ENABLE, DISABLE, ADD and REMOVE statements) and then make this DBDIR the original, you are responsible for its status (attached/detached realms, activated/deactivated online backup capability).

The subschema name must be unique within a multi-DB configuration, i.e. you **cannot** use both the original and shadow database of the same database within a configuration. You can, however, use the original and shadow databases of different databases within a configuration.

The shadow database is handled by RETRIEVAL in the same way as an original database without AFIM logging. Application programs cannot make a distinction between original and shadow databases; they address them via the same subschema names.

If you use an older status of a shadow database, you may need to provide the old user modules, SSITAB modules, etc.

Read-only application programs and read-only UDS/SQL utility routines (e.g. BCHECK, copy routine BOUTLOAD, BPRECORD, BPSIA, BPSQLSIA and BSTATUS) can work with the shadow database concurrently. Of the updating utility routines, only BMEND can access the shadow database. Deactivated realms can also be reconstructed by BMEND in parallel with shadow database operation.

7.3.2 Maintaining ALOG files

Database backups contain the data as it was when the database was saved. In order to be able to reconstruct the most recent status of the data as possible in the event of an error, it is also necessary to record the changes made between backup times. UDS/SQL logs the changes (after-images) made to data in the ALOG file. The ALOG file is database-specific and has the name

dbname.A.*alog-seq-no*

dbname

Name of the database

alog-seq-no

Nine-digit sequence number of the ALOG file

You must specify that changes are to be logged in the ALOG file by using the BMEND utility routine (`START-LOG` /`STOP-LOG` statements) before the start of database operation. If this has not already been done, an ALOG file is created and initialized when `START-LOG` is issued.

You can create the ALOG file yourself by using the command

`CREATE-FILE ...PROTECTION=*PAR(...,USER-ACCESS=*ALL-USERS,...).` This is necessary if the DBH is **not** started under the database user ID, since the DBH cannot create ALOG files under any foreign user ID, but always expects ALOG files under the database user ID on disk.

If AFIM logging is activated, the DBH rejects updating transactions until an ALOG file is available.

Switching the ALOG file

The BMEND utility routine can be used to apply completed ALOG files to a database backup status in order to generate an updated and consistent backup status.

An ALOG file is completed normally only when an ALOG file switch takes place. If AFIM logging is activated, the ALOG file is switched in the following cases:

- The DBH load parameter PP STDCKPT=YES is specified, and the DBH was terminated normally after you performed updates.
- The DBH load parameter PP STDCKPT=YES is specified, and a restart was performed successfully.
- The DBH load parameter PP STDCKPT=YES is specified, and an ALOG file containing a log interval that is not empty is available at the start of the session.
- The DAL command CHECKPOINT is executed.
- An ALOG file overflow occurs.
- An updating utility routine is terminated.
- An updating utility routine is started, and an ALOG file containing a log interval that is not empty is available.
- The BMEND statement STOP-LOG is specified, and an ALOG file containing a log interval that is not empty is available.

> When a database is detached from the current session (`DROP DB`) or when a DBH session is terminated with PP STDCKPT=NO, the ALOG file is not switched and is therefore also not completed.

Regardless of whether or not AFIM is activated, the BMEND statement `START-LOG` switches the ALOG file in the following cases:

- The current ALOG file is not available.
- The current ALOG file already contains a log interval that is not empty.
- The current ALOG file was terminated in an inconsistent state.

Whenever the ALOG file is switched, the internal sequence number of the current ALOG file is incremented by 1.

7.3.3 Reconstructing a database

You must reconstruct a database or parts of it (one or more realms) if, for example, these files fail, are destroyed or are no longer accessible in the original database. Reconstruction involves two steps:

- resetting the database to a backup status
- updating the backup status with ALOG files

The time taken to reconstruct a database depends on how current your backup status is and how much modified data you need to apply.

Resetting to a backup status

The basis for reconstructing the database or parts of it is the database backup. The backup may have been made by

- saving with HSMS or ARCHIVE
- online saving with HSMS or ARCHIVE
- copying the database
- creating a shadow database
- updating a backup status

To reset a database, i.e. generate the original, you must read in the backup once again (HSMS or ARCHIVE) and, if necessary, make it consistent (online saving); alternatively, you may have to copy it back or simply rename it (shadow database).

If you reset the entire database to a consistent status, you can resume database operation immediately as of this backup status. All changes that were made since then, however, will have been canceled. From this time onwards, the ALOG files are overwritten. If you still require the ALOG files, you must copy them beforehand.

If you reset only one realm or several realms, you must update them with ALOG files before you can resume database operation.

Updating the backup status

The BMEND utility routine makes it possible to update the entire database or individual or selected realms using one or more ALOG files. You can use BMEND for the original database, the shadow database and deactivated realms. You apply the ALOG files using the BMEND statement `UPDATE-DATABASE.`

The BMEND statement `SHOW-LOG-INFORMATION` tells you which ALOG files you must use to update the individual realms until you have reached a consistent status for the entire database, i.e. you must make these ALOG files available in order to bring the database up to date.
This statement also tells you whether there is a logging gap in the ALOG file sequence or whether an ALOG file has been terminated in an inconsistent state. In such cases, updating is possible only up to this point or after the database has been saved again.

Applying completed ALOG files

Updating means transferring the modified data from the ALOG file to the database (original or shadow database). A completed ALOG file, i.e. one that has been exchanged for another, is applied in its entirety. You specify the ALOG to which you want to update the database using the DEADLINE parameter of the BMEND statement UPDATE-DATABASE. BMEND automatically ascertains which file is to be used first to update a realm or the database. You must make this and all the other ALOG files up to the file specified in DEADLINE available.

It is also possible to control the application of completed ALOG files via a job variable (`LINK-NAME = JVBMEND`) within a procedure. The BMEND statement `SHOW-LOG-INFORMATION` supplies the job variable with information on the status of the DBDIR and of the log pool, e.g. with the name of the first ALOG file which is to be applied. Once an ALOG file has been applied successfully, the job variable is supplied with the next ALOG file, provided that you have not yet reached the predefined DEADLINE. Further information is provided in the "Recovery, Information and Reorganization" manual under BMEND.

Applying the current ALOG file

It is insufficient in some error situations simply to update to the last consistent status of a database (i.e. to the last completed ALOG file). If, for example, you want to reconstruct a failed realm from an aborted database session, you must also include the changes made to the current ALOG file which, in this case, is inconsistent. You can apply the current ALOG file by specifying DEADLINE=BREAK-POINT in the BMEND statement `UPDATE-DATABASE.`

If the last applied ALOG file is inconsistent, you must next perform a warm start.

Application of the current ALOG file is permitted only on the original database.

Updating a shadow database

The shadow database is updated using the completed ALOG files of the original database. As soon as the ALOG file has been changed for the original database, you can apply the completed ALOG file to the shadow database. This enables you to keep the data of the shadow database very current so that, in the event of an error, the reconstruction time is limited almost to the time taken to apply the current ALOG file.

You can update the shadow database while operation continues on the original database.
You can apply only completed ALOG files.

Updating a deactivated realm

If a hardware error occurs on a database realm, this realm is detached from database operation. A consistency point is set in the database and the ALOG file is switched. When reconstructing a realm, you can read in a backup status of the realm while operation continues on the original database and update it with the relevant ALOG files. You can then reattach the reconstructed, consistent realm to database operation.

ALOG file failure during a warm start

If the exceptional case arises that your ALOG file fails during a warm start due to a hardware error, it is no longer possible to roll back the open transactions on the ALOG file. If this occurs, you can use the BMEND statement KILL-LOG to force deactivation of AFIM logging. This enables you to perform a second warm start **without** AFIM logging, using the RLOG file.

This does, however, cause a gap in logging. To provide a basis for future reconstruction of the database, you should make another full backup of the database once you have performed the warm start (database consistency point) and activated AFIM logging (BMEND statement `START-LOG.`)

7.4 Selecting a recovery concept

To select the recovery concept that meets your own requirements, you must consider various criteria and their relative importance. This section describes some of the concepts available, taking into account the following criteria:

- The update level of the data (e.g. whether data already modified is lost)
- The overhead involved (e.g. memory and storage requirements, cost of making a backup)
- Availability (reconstruction time following an error)

The concepts are listed in increasing order of availability of the database.

No saving

Database operation takes place without any saving or AFIM logging.

Update level
Following an error, all data - both the original data and data that has just been modified - is lost.

Overhead
None.

Availability
None. The database must be generated and loaded again.

Saving the database periodically

The entire database is saved periodically. Database operation takes place without AFIM logging (e.g. to enhance performance). If an error occurs, the database is reset to the last database backup.

Update level
Data that has been modified since the last backup is lost.

Overhead
The database must be saved and archived. If in the event of an error the modified data is to be regenerated, e.g. by repeating the update transaction after saving, the update transactions must be saved separately in the correct chronological order and executed again.

Availability
The availability of the database in the event of an error is very limited. It is determined by the time taken to reset the database to the last backup status and, in some cases, by the time taken to restore data (by repeating the saved transactions). Database operation is not possible while the database is being saved.

Periodic saving and AFIM logging

The database is saved periodically. The changes are recorded in the ALOG file(s) during database operation.

Update level
No data is lost, although the different statuses between backups must be reconstructed.

Overhead

- Creation of the backup (during which time no database operation is possible).
- Archiving of the saved database and the ALOG files.
- Memory and device requirements for AFIM logging.

- Slightly longer duration of transactions due to I/O for AFIM logging.

Availability

The availability of the database in the event of an error is limited.

- Time is required to reset the database to the backup.
- Time is required to update the backup status with ALOG files. The time depends on the number and size of the ALOG files to be applied.
- No database operation is possible while the database is being saved.

Online backup and AFIM logging

Online backup is performed at the backup times during database operation. Database operation is possible only if AFIM logging is activated.

Update level
No data is lost, but intermediate statuses may need to be reconstructed.

Overhead
Memory and device requirements for online saving (depending on the size of the database).

- Archiving of the saved database and the ALOG files.
- Memory and device requirements for AFIM logging.
- Slightly longer duration of transactions due to AFIM logging.

Availability
The availability of the database is increased even more because the backup can be made during database operation.In the event of an error, the availability depends on the time required

- to read in the online backup again
- to make the backup status consistent using ALOG files and to update it, if necessary.

Shadow database and AFIM logging

A shadow database is maintained parallel to the original database. When the ALOG file is switched (e.g. using the DAL command `CHECKPOINT`), the completed ALOG file can be applied to the shadow database immediately.

Update level
The backup has a high update level even in the event of an error; only the data of the current ALOG file must be applied.

Overhead
Creation of the shadow database.

- Memory and device requirements for the shadow database (particularly for large databases).
- Memory and device requirements for AFIM logging.
- Slightly longer duration of transactions due to AFIM logging.
- Immediate updating of the shadow database following an ALOG file switch.

Availability
The availability of the database is very high, even in the event of errors.

The shadow database can be created - by means of online saving - and updated during normal database operation. If errors occur, database operation can be resumed as soon as the defective parts of the database have been replaced (e.g. by renaming the parts of the shadow database) and the current ALOG file has been applied.

7.5 Examples

Example 1

The SHIPPING database is saved online using HSMS. To make this possible, AFIM logging must have been activated and the BMEND statement ENABLE-ONLINE-COPY issued. (Note: In addition to the online backup given in the example, the appropriate ALOG files are required to recover the database.)

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,      VERSION=02.9B00
/START-UDS-BMEND
```

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:26
```

```
//ALLOCATE-BUFFER-POOL BUFFER-SIZE=*STD
//OPEN-DATABASE DATABASE-NAME=SHIPPING
```

```
***** DATABASE ORIGINAL WITHOUT AFIM LOGGING
     FUNCTION ENABLE NOT AVAILABLE
     FUNCTION KILL NOT AVAILABLE
***** CONSISTENT   DATABASE DIRECTORY
```

```
//START-LOG DEFAULT-SUPPORT=*PUBLIC(CATID=*OWN),RESERVE-SUPPORT=*PUBLIC(CATID=ABN2),
                              SPACE=*STD,RESET-LOG-POOL=*NO
```

```
***** LOGGING WILL BE ACTIVATED
     FUNCTION ENABLE AVAILABLE FROM NOW ON
```

```
//ENABLE-ONLINE-COPY
//END
```

```
***** BEGIN       FUNCTION START LOGGING AT 09:27:26
ALOG FILE CREATED ACCORDING TO DEFAULT-SUPPORT
***** NORMAL   END FUNCTION START LOGGING AT 09:27:26
***** BEGIN       FUNCTION ENABLE ONLINE COPY AT 09:27:26
***** ONLINE COPY FOR DATABASE $XXXXXXXX.SHIPPING ALLOWED
***** NORMAL   END FUNCTION ENABLE ONLINE COPY AT 09:27:26
***** DIAGNOSTIC SUMMARY OF BMEND
              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS
***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :         147
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:26
```

```
/START-HSMS
//BACKUP-FILES FILE-NAMES=( -
//   SHIPPING.COSSD -
// , SHIPPING.DBCOM -
// , SHIPPING.DBDIR -
// , SHIPPING.HASHLIB -
// , SHIPPING.ARTICLE-RLM -
// , SHIPPING.CUSTOMER-ORDER-RLM -
// , SHIPPING.PURCHASE-ORDER-RLM -
// , SHIPPING.HOUSEHOLD-GOODS -
// , SHIPPING.CLOTHING -
// , SHIPPING.FOOD -
// , SHIPPING.STATIONERY -
// , SHIPPING.LEISURE -
// , SHIPPING.SPORTS-ARTICLES -
// ), -
//  SAVE-OPTIONS=*PAR(SAVE-ONLINE-FILES=*YES), -
//  ARCHIVE-NAME=UDS.ARC, -
//  OPERATION-CONT=*PAR(OUT=UDS.ARC.SYSLST.BACKUP, -
//                      REPORT=*FULL,WAIT-FOR-COMPLETION=*YES)
// END
```

```
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=BCF#4TKB REQUEST-DATE=2019-01-29 09:27:29 USER-ID=XXXXXXXX  REQUEST-
STATE=COMPLETED WITHOUT ERROR
STATEMENT LISTING:
BACKUP-FILES   FILE-NAMES = ( SHIPPING.COSSD , SHIPPING.DBCOM , SHIPPING.DBDIR , SHIPPING.
HASHLIB , SHIPPING.ARTICLE-RLM , SHIPPING.CUSTOMER-ORDER-RLM , SHIPPING.PURCHASE-ORDER-RLM ,
SHIPPING.HOUSEHOLD-GOODS , SHIPPING.CLOTHING , SHIPPING.FOOD ,  SHIPPING.STATIONERY ,
SHIPPING.LEISURE , SHIPPING.SPORTS-ARTICLES ),
SAVE-OPTIONS = *PAR(SAVE-ONLINE-FILES = *YES), ARCHIVE-NAME = UDS.ARC, OPERATION-CONT=*PAR
(OUT = UDS.ARC.SYSLST.BACKUP, REPORT = *FULL, WAIT-FOR-COMPLETION=*YES)

               ENVIRONMENT                          : SF
               ARCHIVE-NAME                         : $XXXXXXXX.UDS.ARC
               SAVE-FILE ATTRIBUTES                 : NEW
               TO-STORAGE                           : S2-STORAGE-LEVEL
               DEVICE-TYPE                          : TAPE-C4
               RETENTION-PERIOD                     : 30
               SAVE-VERSION ATTRIBUTES
               SAVE-VERSION-NAME                    :
               SELECT-FILES PARAMETER
               MODIFIED-FILES  PARTIAL-FILE-SAVE    : NO
               MAX-BACKUP-CLASS                     : STD
***   BACKUP - FILES              HSMS V11.0       FULL        REPORT  *** 2019-01-29  09:
27:37     PAGE    2
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=BCF#4TKB REQUEST-DATE=2019-01-29 09:27:29 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITHOUT ERROR


%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.190129.092730', VERSION '11.0A04'
%  ARC0033 ARCHIVE SUBTASK TSN '4TKL' GENERATED
%  ARC0825 SUBTASK '0' HAS TRANSFERRED '1478' PAM PAGES FOR '13' FILES AND '0' JVS WITH 256K-
```

```
BLOCKING IN '1' SECONDS
  SAVE FILE IDENTIFIER - S.190129.092730    SAVE-VERSION-DATE=2019-01-29  SAVE-VERSION-
TIME=09:27:30
          SUBSAVE
          NUMBER       VSNS

             0        FI0994

   SAVE FILE IDENTIFIER - S.190129.092730     SAVE-VERSION-DATE=2019-01-29   SAVE-VERSION-
TIME=09:27:30
                    ***   CATALOG - IUDS      USER - XXXXXXXX   ***
          FILE/JOB VARIABLE NAME                      LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                                     VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)

SHIPPING.ARTICLE-RLM                                    1      124  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.CLOTHING                                       1      108  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.COSSD                                          1       12  FULL  IUDS.2  D     0  FI0994
SHIPPING.CUSTOMER-ORDER-RLM                             1       72  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.DBCOM                                          1      500  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.DBDIR                                          1      200  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.FOOD                                           1       36  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.HASHLIB                                        1        9  FULL  IUDS.6  D     0  FI0994
SHIPPING.HOUSEHOLD-GOODS                                1       48  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.LEISURE                                        1       88  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.PURCHASE-ORDER-RLM                             1      120  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.SPORTS-ARTICLES                                1       88  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.STATIONERY                                     1       48  FULL  IUDS.2  D     0  FI0994
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
    NUMBER OF FILES=            13 GLOBAL SIZE=        1453 START= 2019-01-29 09:27:29 END=
2019-01-29 09:27:37
***      E N D   O F                HSMS V11.0        FULL            REPORT  *** 2019-01-
29  09:27:37     ***
```

Example 2

The online backup of the SHIPPING database is read in using HSMS. The status of the backup (consistency, log interval) is queried using BMEND.
The database is then updated to the last possible consistency point with BMEND.

```
/START-HSMS
//RESTORE-FILES -
// FILE-NAMES=(SHIPPING.),-
// ARCHIVE-NAME=UDS.ARC,-
// REPLACE-FILES=YES,-
// SELECT-SAVE-VERSIONS=*LATEST,-
// OPERATION-CONT=*PAR(OUT=UDS.ARC.SYSLST.RESTORE, -
//                     REPORT=*FULL,WAIT-FOR-COMPLETION=*YES)
//STEP
//END
```

```
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=RSF#4TTF REQUEST-DATE=2019-01-29 09:27:44 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITHOUT
ERROR

STATEMENT LISTING:

RESTORE-FILES   FILE-NAMES=(SHIPPING.),    ARCHIVE-NAME=UDS.ARC,   REPLACE-FILES=YES,
SELECT-SAVE-
VERSIONS=*LATEST,
               OPERATION-CONT=*PAR(OUT=UDS.ARC.SYSLST.RESTORE, REPORT=*FULL,WAIT-FOR-
COMPLETION=*YES)

%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.190129.092744', VERSION '11.0A04'
%  ARC0033 ARCHIVE SUBTASK TSN '4TK4' GENERATED
%  ARC0815 SUBTASK '0' HAS TRANSFERRED '1478' PAM PAGES FOR '13' FILES AND '0' JVS IN '0'
SECONDS

                    SAVE FILE IDENTIFIER - S.190129.092730
          SUBSAVE
          NUMBER      VSNS

             0       FI0994

                   ***   CATALOG - IUDS      USER - XXXXXXXX   ***
          FILE/JOB VARIABLE NAME               LASTPG/   SAVE VERSION  SAVE  INPUT   SUB
OUTPUT
                                          VERS     SIZE    IDENTIFIER   TYPE   VSN   SAVE DISK
(S)
SHIPPING.ARTICLE-RLM                         1     124  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.CLOTHING                            1     108  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.COSSD                               1      12  190129.092730  FULL  FI0994   0
IUDS.2
```

```
SHIPPING.CUSTOMER-ORDER-RLM                1      72  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.DBCOM                             1     500  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.DBDIR                             1     200  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.FOOD                              1      36  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.HASHLIB                           1       9  190129.092730  FULL  FI0994   0
IUDS.6
SHIPPING.HOUSEHOLD-GOODS                   1      48  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.LEISURE                           1      88  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.PURCHASE-ORDER-RLM                1     120  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.SPORTS-ARTICLES                   1      88  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'
SHIPPING.STATIONERY                        1      48  190129.092730  FULL  FI0994   0
IUDS.2
%  ARC0059 FILE HAS BEEN SAVED 'ON LINE'



***     E N D   O F      HSMS V11.0      FULL         REPORT  *** 2019-01-29  09:27:37    ***
```

/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.9B00
/START-UDS-BMEND

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:46
```

//SHOW-LOG-INFORMATION REALM-NAME=*ALL,LOG-FILE=4,OUTPUT=*SYSOUT

```
***** DATABASE ORIGINAL WITHOUT AFIM LOGGING
      FUNCTION ENABLE NOT AVAILABLE
      FUNCTION KILL NOT AVAILABLE
***** CONSISTENT   DATABASE DIRECTORY
```

//END

```
***** BEGIN       FUNCTION SHOW LOG INFORMATION AT 09:27:49
***** LOG INFORMATION FOR DATABASE $XXXXXXXX.SHIPPING
***** LOG INTERVAL OF SPECIFIED REALMS
                            !     ALOG SEQ NR     !            !
REALM-NAME                  !  BEGIN   !  END     ! CONSISTENT !
----------------------------+----------+----------+------------+
DATABASE-DIRECTORY          !        2 !        2 !    NO      !
DATABASE-COMPILER-REALM     !        1 !        1 !    YES     !
CUSTOMER-ORDER-RLM          !        1 !        1 !    YES     !
PURCHASE-ORDER-RLM          !        2 !        2 !    NO      !
CLOTHING                    !        1 !        1 !    YES     !
HOUSEHOLD-GOODS             !        1 !        1 !    YES     !
SPORTS-ARTICLES             !        1 !        1 !    YES     !
FOOD                        !        2 !        2 !    NO      !
LEISURE                     !        1 !        1 !    YES     !
STATIONERY                  !        1 !        1 !    YES     !
ARTICLE-RLM                 !        2 !        2 !    NO      !
***** LOG MODE : NO LOGGING
***** SUPPORTS OF ACTUAL LOG FILE :
***** INFORMATION ABOUT LOG HISTORY :



 ALOG SEQ NR!      LOG INTERVAL         ! AFIM !BACKOUT! LOGGING !
          *--------------*--------------*      !        !         !
          !     BEGIN    !     END      !      ! BFIM  !   GAP   !
----------*--------------*--------------*------*-------*---------*
       4  !20190129092738!20190129092739!   *  !       !    ?    !
       3  !20190129092737!20190129092738!   *  !       !         !
       2  !20190129092727!20190129092737!   *  !       !         !
       1  !20190129092725!20190129092727!   *  !       !         !
***** ONLINE COPIES BY ARCHIVE ARE  NOT ALLOWED
***** NORMAL   END FUNCTION SHOW LOG INFORMATION AT 11:41:01

***** DIAGNOSTIC SUMMARY OF BMEND

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          51
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:49
```

/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,   VERSION=02.9B00
/START-UDS-BMEND
...

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:49
```

//OPEN-DATABASE DATABASE-NAME=SHIPPING
//UPDATE-DATABASE REALM-NAME=*ALL,DEADLINE=*STD

```
SYSTEM_BREAK OCCURRED IN REALM DATABASE-DIRECTORY
***** INCONSISTENT DATABASE DIRECTORY
SYSTEM_BREAK OCCURRED IN REALM PURCHASE-ORDER-RLM
SYSTEM_BREAK OCCURRED IN REALM FOOD
SYSTEM_BREAK OCCURRED IN REALM ARTICLE-RLM
***** INCONSISTENT DATABASE DIRECTORY
      FUNCTION ADD NOT AVAILABLE
      FUNCTION REMOVE NOT AVAILABLE
      FUNCTION START NOT AVAILABLE
      FUNCTION STOP NOT AVAILABLE
***** STD UPDATE FOR ALL REALMS SPECIFIED
      FUNCTION ADD AVAILABLE FROM NOW ON
      FUNCTION REMOVE AVAILABLE FROM NOW ON
      FUNCTION START AVAILABLE FROM NOW ON
      FUNCTION STOP AVAILABLE FROM NOW ON
      FUNCTION KILL NOT AVAILABLE
```

//END

```
***** BEGIN        FUNCTION UPDATE DATABASE AT 09:27:49
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000002' STARTED
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000002' FINISHED
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000003' STARTED
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000003' FINISHED
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000004' STARTED
***** MENDING WITH ALOG FILE '$XXXXXXXX.SHIPPING.A.000000004' FINISHED
***** ALOG FILE '$XXXXXXXX.SHIPPING.A.000000005' NOT USED FOR MENDING
***** NORMAL   END FUNCTION UPDATE DATABASE AT 09:27:49

***** DIAGNOSTIC SUMMARY OF BMEND

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS
***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          306
***** NORMAL END   BMEND        (UDS/SQL  V2.9  1801 )     2019-01-29   09:27:49
```

Example 3

A shadow database CUSTOMER.COPY is generated from the CUSTOMER database.

/SHOW-FILE-ATTRIBUTES FILE-NAME=CUSTOMER.

```
      768 :IUDS:$XXXXXXXX.CUSTOMER.A.000000001
      192 :IUDS:$XXXXXXXX.CUSTOMER.A.000000002
      120 :IUDS:$XXXXXXXX.CUSTOMER.COSSD
      252 :IUDS:$XXXXXXXX.CUSTOMER.CUSTOMER-RLM
      501 :IUDS:$XXXXXXXX.CUSTOMER.DBCOM
      201 :IUDS:$XXXXXXXX.CUSTOMER.DBDIR
       24 :IUDS:$XXXXXXXX.CUSTOMER.DBSTAT
       24 :IUDS:$XXXXXXXX.CUSTOMER.DBSTAT.SAVE
      252 :IUDS:$XXXXXXXX.CUSTOMER.FINANCE-RLM
       12 :IUDS:$XXXXXXXX.CUSTOMER.HASHLIB
       24 :IUDS:$XXXXXXXX.CUSTOMER.SLF
IUDS PUBLIC:     11 FILES RES=     2370 FRE=      351 REL=      339 PAGES
```

/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.COSSD, TO-FILE=CUSTOMER.COSSD.COPY
/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.DBCOM, TO-FILE=CUSTOMER.DBCOM.COPY
/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.DBDIR, TO-FILE=CUSTOMER.DBDIR.COPY
/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.HASHLIB, TO-FILE=CUSTOMER.HASHLIB.COPY
/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.CUSTOMER-RLM, TO-FILE=CUSTOMER.CUSTOMER-RLM.COPY

/COPY-FILE FROM-FILE=:IUDS:CUSTOMER.FINANCE-RLM, TO-FILE=CUSTOMER.FINANCE-RLM.COPY
/SHOW-FILE-ATTRIBUTES FILE-NAME=CUSTOMER.*.COPY

```
      120 :IUDS:$XXXXXXXX.CUSTOMER.COSSD.COPY
      252 :IUDS:$XXXXXXXX.CUSTOMER.CUSTOMER-RLM.COPY
      501 :IUDS:$XXXXXXXX.CUSTOMER.DBCOM.COPY
      201 :IUDS:$XXXXXXXX.CUSTOMER.DBDIR.COPY
      252 :IUDS:$XXXXXXXX.CUSTOMER.FINANCE-RLM.COPY
       12 :IUDS:$XXXXXXXX.CUSTOMER.HASHLIB.COPY
:IUDS: PUBLIC:     6 FILES RES=     1338 FRE=      119 REL=      117 PAGES
```

Example 4

The defective realm CUSTOMER.CUSTOMER-RLM is replaced by the realm backup CUSTOMER.CUSTOMER-REALM.COPY. The status (consistency, log interval) of the realm is queried using BMEND. The realm is then updated to the last possible consistency point with BMEND.

```
/DELETE-FILE :IUDS:CUSTOMER.FINANCE-RLM
/COPY-FILE FROM-FILE=CUSTOMER.FINANCE-RLM.COPY    ,TO-FILE=:IUDS:CUSTOMER.FINANCE-
RLM
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.9B00
/START-UDS-BMEND
...
```

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

```
//OPEN-DATABASE DATABASE-NAME=CUSTOMER
```

```
DIFFERENCE OF BACKUP DATA BETWEEN REALM FINANCE-RLM AND CONSISTENCY RECORD
***** INCONSISTENT DATABASE DIRECTORY
      FUNCTION START NOT AVAILABLE
      FUNCTION STOP NOT AVAILABLE
```

```
//SHOW-LOG-INFORMATION REALM-NAME=*ALL,OUTPUT=*SYSLST
//END
```

```
***** BEGIN       FUNCTION SHOW LOG INFORMATION AT 09:26:43

          ***** LOG INFORMATION FOR DATABASE $XXXXXXXX.CUSTOMER

          ***** LOG INTERVAL OF SPECIFIED REALMS
                          !     ALOG SEQ NR      !            !
REALM-NAME                !  BEGIN    !  END     ! CONSISTENT !
--------------------------+-----------+----------+------------+
DATABASE-DIRECTORY        !         3 !        3 !    YES     !
DATABASE-COMPILER-REALM   !         2 !        2 !    YES     !
CUSTOMER-RLM              !         2 !        2 !    YES     !
FINANCE-RLM               !         1 !        1 !    YES     !
          ***** TO MAKE THE SPECIFIED REALMS CONSISTENT, THE FOLLOWING LOG FILES ARE
                NECESSARY :
               FROM ALOG SEQ NR        1 TO ALOG SEQ NR        3
               OR FROM LOG INTERVAL BEGIN 20190129092640 TO LOG INTERVAL END
                 20190129092642

          ***** LOG MODE : AFIM LOGGING

          ***** SUPPORTS OF ACTUAL LOG FILE :

     DEFAULT SUPPORT :PVS ID = DEFAULT PVS
     RESERVE SUPPORT :PVS ID = DEFAULT PVS

***** INFORMATION ABOUT LOG HISTORY :


ALOG SEQ NR!       LOG INTERVAL         ! AFIM !BACKOUT! LOGGING !
          *--------------*--------------*      !       !         !
          !     BEGIN    !      END     !      !  BFIM !   GAP   !
----------*-------------*--------------*------*-------*---------*
       3  !20190129092642!20190129092642!   *  !       !         !
       2  !20190129092640!20190129092642!   *  !       !         !
       1  !20190129092639!20190129092640!   *  !       !         !

           ***** ONLINE COPIES BY ARCHIVE ARE  NOT ALLOWED

***** NORMAL   END FUNCTION SHOW LOG INFORMATION AT 09:26:43

***** DIAGNOSTIC SUMMARY OF BMEND

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          20
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:42
```

/START-UDS-BMEND

...

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

//OPEN-DATABASE DATABASE-NAME=CUSTOMER

```
DIFFERENCE OF BACKUP DATA BETWEEN REALM FINANCE-RLM AND CONSISTENCY RECORD
***** INCONSISTENT DATABASE DIRECTORY
      FUNCTION START NOT AVAILABLE
      FUNCTION STOP NOT AVAILABLE
```

//UPDATE-DATABASE REALM-NAME=FINANCE-RLM,DEADLINE=*STD
//END

```
***** BEGIN       FUNCTION UPDATE DATABASE AT 09:26:43
***** ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000001' NOT USED FOR MENDING
***** MENDING WITH ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000002' STARTED
***** MENDING WITH ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000002' FINISHED
***** ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000003' NOT USED FOR MENDING
***** NORMAL   END FUNCTION UPDATE DATABASE AT 09:26:43

***** DIAGNOSTIC SUMMARY OF BMEND


             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS


***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          41
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

Example 5

The shadow database CUSTOMER.COPY is to be updated. Its status (consistency, log interval) compared to the original database is queried using BMEND. The shadow database is then updated to the last possible consistency point with BMEND.

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,      VERSION=02.9B00
/START-UDS-BMEND
...
***** START       BMEND       (UDS/SQL  V2.9  1801 )    2019-01-29   09:26:44


//OPEN-DATABASE DATABASE-NAME=CUSTOMER,COPY-NAME=COPY
***** PROCESSING OF DATABASE COPY
      FUNCTION START NOT AVAILABLE
      FUNCTION STOP NOT AVAILABLE
      FUNCTION KILL NOT AVAILABLE
***** CONSISTENT   DATABASE DIRECTORY

//SHOW-LOG-INFORMATION REALM-NAME=*ALL,OUTPUT=*SYSLST
//END
```

```
***** BEGIN       FUNCTION SHOW LOG INFORMATION AT 09:26:44

          ***** LOG INFORMATION FOR DATABASE $XXXXXXXX.CUSTOMER.COPY

          ***** LOG INTERVAL OF SPECIFIED REALMS
                              !     ALOG SEQ NR      !            !
REALM-NAME                    !  BEGIN    !  END     ! CONSISTENT !
------------------------------+-----------+---------+------------+
DATABASE-DIRECTORY            !         2 !        2 !    YES     !
DATABASE-COMPILER-REALM       !         1 !        1 !    YES     !
CUSTOMER-RLM                  !         1 !        1 !    YES     !
FINANCE-RLM                   !         1 !        1 !    YES     !

***** INFORMATION ABOUT LOG HISTORY :

ALOG SEQ NR!       LOG INTERVAL          ! AFIM !BACKOUT! LOGGING !
          *--------------*--------------*     !        !         !
          !     BEGIN    !     END      !     !  BFIM  !   GAP   !
-----------*-------------*--------------*------*-------*---------*
       2  !20190129092640!20190129092640!   *  !       !    ?    !
       1  !20190129092639!20190129092640!   *  !       !         !

           ***** ONLINE COPIES BY ARCHIVE ARE  NOT ALLOWED

***** NORMAL   END FUNCTION SHOW LOG INFORMATION AT 09:26:40
***** DIAGNOSTIC SUMMARY OF BMEND

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY

***** NR OF DATABASE ACCESSES  :          20
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:40
```

/START-UDS-BMEND

...

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

//OPEN-DATABASE DATABASE-NAME=CUSTOMER

```
***** CONSISTENT   DATABASE DIRECTORY
      FUNCTION KILL NOT AVAILABLE
```

//SHOW-LOG-INFORMATION REALM-NAME=*ALL,OUTPUT=*SYSLST

//END

```
***** BEGIN       FUNCTION SHOW LOG INFORMATION AT 09:26:43

          ***** LOG INFORMATION FOR DATABASE $XXXXXXXX.CUSTOMER

          ***** LOG INTERVAL OF SPECIFIED REALMS

                           !     ALOG SEQ NR      !            !
REALM-NAME                 !  BEGIN    !  END     ! CONSISTENT !
---------------------------+-----------+----------+------------+
DATABASE-DIRECTORY         !         3 !        3 !    YES     !
DATABASE-COMPILER-REALM    !         2 !        2 !    YES     !
CUSTOMER-RLM               !         2 !        2 !    YES     !
FINANCE-RLM                !         2 !        2 !    YES     !

          ***** LOG MODE : AFIM LOGGING

          ***** SUPPORTS OF ACTUAL LOG FILE :

     DEFAULT SUPPORT :PVS ID = DEFAULT PVS
     RESERVE SUPPORT :PVS ID = DEFAULT PVS

***** INFORMATION ABOUT LOG HISTORY :

ALOG SEQ NR!       LOG INTERVAL        ! AFIM !BACKOUT! LOGGING !
           *--------------*------------*     !        !         !
           !     BEGIN    !     END    !     !  BFIM  !   GAP   !
-----------*--------------*------------*-----*--------*---------*
       3  !20190129092642!20190129092642! *  !      !        !
       2  !20190129092640!20190129092642! *  !      !        !
       1  !20190129092639!20190129092640! *  !      !        !


          ***** ONLINE COPIES BY ARCHIVE ARE  NOT ALLOWED

***** NORMAL   END FUNCTION SHOW LOG INFORMATION AT 09:26:43
***** NR OF DATABASE ACCESSES  :          20
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

/ START-UDS-BMEND

...

```
***** START       BMEND       (UDS/SQL  V2.9  1801 )     2019-01-29   09:26:43
```

//OPEN-DATABASE DATABASE-NAME=CUSTOMER,COPY-NAME=COPY

```
***** PROCESSING OF DATABASE COPY
      FUNCTION START NOT AVAILABLE
      FUNCTION STOP NOT AVAILABLE
      FUNCTION KILL NOT AVAILABLE
***** CONSISTENT   DATABASE DIRECTORY
```

//UPDATE-DATABASE REALM-NAME=*ALL,DEADLINE=*STD
//END

```
***** BEGIN       FUNCTION UPDATE DATABASE AT 09:26:43
***** MENDING WITH ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000002' STARTED
***** MENDING WITH ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000002' FINISHED
***** ALOG FILE '$XXXXXXXX.CUSTOMER.A.000000003' NOT USED FOR MENDING
***** NORMAL   END FUNCTION UPDATE DATABASE AT 09:26:43
***** DIAGNOSTIC SUMMARY OF BMEND

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :         147
***** NORMAL END   BMEND       (UDS/SQL  V2.9  1801 )    2019-01-29   09:26:43
```

8 Optimizing performance

In order to improve the behavior and performance of UDS/SQL, it is essential that the databases being used are structured appropriately for the applications involved. A number of facilities are available in UDS/SQL both for the logical and the physical organization of data, and thus for optimizing access to data stored in the database (see the "Design and Definition" manual for details).

- One method of increasing performance substantially is to avoid unnecessary I/O operations when accessing data stored in databases. A description of how I/O behavior can be analyzed and controlled can be found in section “Optimizing I/O behavior”.
- The amount of working memory required for the UDS/SQL configuration can be reduced by the concurrent sharing of programs that are loaded as subsystems. A description of the various subsystems that can be created for UDS/SQL is provided in section “Optimizing usage of working memory with the subsystem functionality of UDS/SQL”.
- In the case of multi-processor systems, the performance of the independent DBH can be improved dramatically by the dynamic distribution of UDS/SQL tasks to different processors. The methods used to analyze and control processor utilization are dealt with in section “Optimizing processor utilization with the independent DBH”.

8.1 Optimizing I/O behavior

In the case of large database applications, I/O behavior plays a crucial role in the performance of the application. A description of the files in which I/O bottlenecks may occur and how such bottlenecks can be eliminated is provided below.

Session Log File (SLF)

No I/O bottleneck can occur for the SLF.

Status file

No I/O bottleneck can occur for the status file.

Realms

An I/O bottleneck for realms can be eliminated by many methods:

- Increase the database buffer

  Increasing the database buffer reduces the number of I/Os and can thus eliminate the bottleneck. The amount of main memory available should, however, also be taken into account to ensure that the corresponding increase in the paging rate is acceptable.

  It is, of course, also important that the correct database buffer (2-Kbyte, 4-Kbyte or 8-Kbyte) be increased, so you will first need to determine the appropriate database page size.

  The UDS/SQL monitor can be used to check whether increasing the database buffer has reduced the number of physical I/Os.

- Create a separate database buffer for a database

  If particular database is critical for performance, you can create a separate database buffer for it. This will ensure that the pages of that database are not displaced by I/Os on other databases, but also means that the dedicated buffer will not be available for I/Os on other databases.

- Distribute realms to different disks

  If increasing the database buffer does not have the desired effect, you can distribute the realms on different disks to reduce the number of I/Os on a disk and thus eliminate the I/O bottleneck.

- Use a disk cache or a solid state disk (SSD)

  If the I/O bottleneck cannot be eliminated by increasing the database buffer or by distributing realms on different disks, you can use a disk cache for accelerated I/O or some other external fast storage medium for more demanding requirements.

- Modify the application

  In some cases, an I/O bottleneck may occur due to the inefficient sequence of database calls in an application. Updating the application from this viewpoint could reduce the number of I/Os substantially; however a detailed analysis of the application is required for this purpose.

RLOG file

Every updating transaction requires a write operation on the RLOG file. Although multiple transactions can be committed with a single I/O operation (group commit), the RLOG file could become a bottleneck in applications with several updating transactions. The following solutions are available in such cases:

- Place the RLOG file on a separate disk

  The RLOG file is written sequentially. You should create the RLOG file in such a manner that writing can take place as quickly as possible and without being influenced by other activities on the system.
- Use a disk cache or an SSD

  If the bottleneck cannot be eliminated with measure indicated above, it is advisable to use a disk cache or a fast external storage medium.
- Use a global storage (GS)

  If the bottleneck cannot be removed with an SSD either, the pubset on which the RLOG file is located can be buffered via a global storage. If the RLOG file is mirrored, to Availability Units of the Global Storage will be required in order the guarantee the independent redundancy of the two RLOG files.

ALOG file

The ALOG file is generally not critical for performance, since it is only written when the ALOG buffer is full.

If the ALOG file does become a bottleneck, however, it is best to place it on a disk on which no other access occurs. You should then create the ALOG file in such a manner that writing can take place as quickly as possible and without being influenced by other activities on the system.

Temporary realms

Temporary realms are generally not critical for performance.

However, if they do slow down performance, you can try increasing the 4-Kbyte system buffer (since temporary realms always have a page size of 4 Kbytes). If this does not help either, you will need to distribute the temporary realms on multiple disks.

8.2 Optimizing usage of working memory with the subsystem functionality of UDS/SQL

In order to support the optimum use of working memory, UDS/SQL is supplied as a subsystem. This includes object module files for the UDS-SQL and UDS-D subsystems.

The UDS-SQL subsystem includes the shareable code that is needed to run UDS/SQL locally. It can generally be used in combination with linked-in and independent applications.

The UDS-D subsystem includes the shareable code required for the supplementary use of UDS-D.

The subsystems can be incorporated in an existing or supplementary subsystem catalog by the BS2000 system administrator by using subsystem management resources (see the “Subsystem Management in BS2000” manual for details).

If UDS-SQL or UDS-D was installed with IMON, the user ID under which the subsystems were installed will be known to the system.

If UDS-SQL or UDS-D was not installed with IMON, the subsystems are searched under the default user ID (usually $TSOS). If they are installed under some other user ID, the BS2000 system administrator must copy the library SYSLNK.UDS-SQL.029 or SYSLNK.UDS-D.029 to the default user ID or change the `INSTALLATION-` parameter (see below) to match the installation user ID.

The BS2000 system administrator must add the subsystem to the subsystem configuration (DSSM command `ADD-SUBSYSTEM`). From now on, assignment to this subsystem is possible via the BS2000 command SELECT-PRODUCT-VERSION.

With the BS2000 command `MODIFY-SUBSYSTEM-PARAMETER`, the BS2000 system administrator can modify the attributes of the subsystem. However, only changes to the following operands are advisable:

INSTALLATION-USERID

If UDS-SQL was installed with IMON, the installation user ID is used.

If UDS-SQL was not installed with IMON, the default user ID (usually $TSOS) is used. In this case, the LLM modules of the subsystem must be available under the default user ID in the library from which the subsystem is to be loaded (LIBRARY parameter).

If the subsystem is to be loaded from some other user ID, the BS2000 system administrator can also set that user ID by means of the INSTALLATION-USERID parameter.

LIBRARY

Default value: `SYSLNK.UDS-SQL.`*nnn* `(`*nnn*`=version)`
Designates the library from which the subsystem is to be loaded.

MEMORY-CLASS

Default value: `*SYSTEM-GLOBAL`
Loads the subsystem into class-3/class-4 memory. The use of class-5 memory (LOCAL-PRIVILEGED) is advantageous only in exceptional cases (see the section “Memory and task concept” in the "Subsystem Management in BS2000" manual).

SUBSYSTEM-ACCESS

Default value: *HIGH

Loads the subsystem above 16 Mbytes so that it can be used by all XS-compatible applications and UDS /SQL tasks.

If the subsystem is to be used by non-XS-compatible applications, the BS2000 system administrator can also load the subsystem below 16 Mbytes (`SUBSYSTEM-ACCESS=*LOW`). This is, however, subject to certain restrictions (see the corresponding notes in the “Memory and task concept” section of the "Subsystem Management in BS2000" manual).

> In some cases, an attempt is initially made in application programs to load the UDS/SQL connection modules below 16 Mbytes. If the subsystem has been loaded high (with the parameter `SUBSYSTEM-ACCESS=*HIGH`), this could result in the following error message for each dynamically loaded connection module:
>
> ```
> BLS0061 AMODE OF SYMBOL ’....’ INVALID IN THIS ENVIRONMENT.
> LOADING ABORTED.
> ```
>
> UDS/SQL loads the UDS/SQL connection module again with modified parameters. If no further errors are reported, these messages from the Binder-Loader system may be ignored.

The subsystem is activated by the BS2000 system administrator with the DSSM command START-SUBSYSTEM. If several subsystem versions are to be active in parallel, the following specification must be made in the `START-SUBSYSTEM` command: `VERSION-PARALLELISM=*COEXISTENCE-MODE`

8.3 Optimizing processor utilization with the independent DBH

One of the main objectives when using UDS/SQL is to ensure that existing processors are utilized efficiently. This is advantageous not only to the UDS/SQL user, who directly benefits from the quicker response times and increased throughput, but also to all other applications and programs on the system, since the available CPU resources are used economically.

The UDS/SQL server tasks of the independent DBH accept requests from application tasks without any fixed association. Requests are maintained in a common request pool, from which they are retrieved equally by all servers and processed. Requests that have not been fully processed because they are waiting for the completion of an I/O, for example, are returned to the same pool and can then be completed by any server. No load control and distribution is required by the database administrator.

Every server task uses “multithreading”, i.e. interrupts any request that cannot be processed further, returns the request, and begins a new one. This method ensures high processor availability for the server tasks and thus guarantees excellent performance.

8.3.1 Load parameters that affect processor utilization

The following DBH load parameters can be used to control processor utilization:

- PP DEACT
- PP RESULT-DELAY
- PP SCHEDULING
- PP SERVERTASK
- PP WAIT

The load parameter PP DEACT

The load parameter DEACT can be used to control how server tasks are to be handled by the operating system in high-load situations.

If DEACT=NO is set, server tasks are not affected much by other tasks in the system and thus operate more efficiently even in high-load situations where a high percentage of system resources are used and in wait states. This could, however, have an adverse effect on other applications.

The load parameter PP RESULT-DELAY

The load parameter PP RESULT-DELAY can be used to group request results for user tasks and thus improve throughput in high-load situations.

Note, however, that if PP SCHEDULING=ASYMMETRIC has also been set, the number of request results actually grouped may be restricted.

The load parameter PP SCHEDULING

The load parameter PP SCHEDULING can be used to control the internal behavior of multiple server tasks in peak-load situations with a high degree of fluctuation.

PP SCHEDULING=SYMMETRIC causes all server tasks to process requests equally. It is useful for well-utilized server tasks or in cases where no high BS2000 priority can be assigned to the server tasks.

This setting is the default value, since the best performance in cases where loads fluctuate dynamically is obtained by having all requests processed equally by the server tasks, i.e. by a setting appropriate for the median load.

PP SCHEDULING=ASYMMETRIC causes the load on the server tasks to be unequal. This allows the processor cache to be used more efficiently and also enables better coordination with the task scheduling of the operating system, especially in cases when the server tasks cannot be fully utilized, but where even one server task less would be insufficient.

The load parameter PP SERVERTASK

In order to utilize the processors efficiently, it is essential that the load parameter PP SERVERTASK be set correctly. Appropriate consideration must be given to BS2000 priorities to achieve the optimum effect for the setting (see the section “Performanceoriented BS2000 settings” for details).

Due to the multi-threading facility, a single server task (PP SERVERTASK=1) is usually sufficient on a monoprocessor when using asynchronous I/O (PP IO=ASYNC). The use of additional server tasks is only advantageous on multiprocessor systems. The upper limit for the load parameter PP SERVERTASK is usually (i.e. with PP IO=ASYNC) the number of processors.

Note, however, that a higher value is also accepted under certain circumstances, i.e. if the load parameter PP IO=SYNC is specified.

The appropriate setting for the load parameter PP SERVERTASK can be determined for a normal situation (PP IO=ASYNC) with a high user load by means of a simple measurement. This is done by setting the load parameter PP SERVERTASK to the possible upper limit in the measurement configuration and then determining the CPU utilization of the server tasks, which can be detected from the program name UDSSUB. The measurement for CPU utilization can be performed easily at the start and end of a measurement phase by using the BS2000 command SHOW-USER-STATUS.

If you add the CPU utilization of tasks with the program UDSSUB (detectable with SHOW-USER-STATUS INFORMATION=PROGRAM) and then set this value in relation to the CPU utilization of all active tasks in the system, you will obtain a good approximation of the proportion of CPU resources taken up by server tasks in each measurement phase.

In order to determine the “correct” value for the load parameter PP SERVERTASK, you can then multiply the maximum value across all measurement phases with the number of processors on the system.

For example, if the server tasks collectively use 41% of CPU resources on a quadprocessor system, you will need 1.64 (=0.41 x 4) server tasks. The appropriate setting would then be PP SERVERTASK=2.

If a configuration is subject to a high fluctuation in load, the setting for PP SERVERTASK can be adjusted to suit the requirements for the peak load (i.e. the best response times) or an average load (i.e. optimum CPU utilization during normal operation).

Note that setting the load parameter PP SERVERTASK too high will increase task communication significantly, i.e. result in higher CPU utilization and thus possibly impede other applications.

A setting that is too low, by contrast, will generally reduce the throughput rate in the UDS/SQL configuration.

The load parameter PP WAIT

The load parameter PP WAIT can be used to set an active waiting strategy.

Under normal circumstances, if the load is low, waiting for new requests to arrive is controlled via eventing. This ensures that the CPU can also be used by other applications if no further requests can be processed by the server tasks.

If it is possible to ensure through the application that such “underload” situations are extremely rare or very brief, the CPU overhead required for eventing in the normal and peak-load case can be reduced by active waiting.

> The load parameters PP RESULT-DELAY and PP WAIT are intended for peak-load configurations and should only be used if the conditions described for these load parameters are actually fulfilled (see RESULT-DELAY in chapter "DBH load parameters" and WAIT in chapter "DBH load parameters"). Otherwise, they could have a strong adverse effect on operations.

8.3.2 Performance-oriented BS2000 settings

In order to ensure efficient CPU utilization in a UDS/SQL configuration, it is essential that the correct BS2000 settings be used. These settings are described below.

Task priorities

BS2000 offers a priority facility to control system behavior in peak and overload situations (see the manual "Introduction to System Administration" for details).

The best method of ensuring guaranteed behavior for server tasks over an extended period is to use fixed priorities for peak-load requirements. The permanent prioritization of server tasks achieved by this method could, however, have a negative impact on other tasks of the same category.

Some settings of load parameters are only meaningful in combination with fixed priorities.

The effect of various settings are explained in section “Example of a peak-load configuration”.

Task attribute TP

UDS/SQL switches every server task to the BS2000 task attribute TP, which enables continuous utilization of processors by server tasks. Note, however, that the job class which uses the server tasks must be assigned to the task category TP (see the job class assignment on "Starting DBH").

The BS2000 command `SHOW-JOB-CLASS` can be used to verify whether the task category TP has been assigned for the job class. The job class assignment can be checked with the BS2000 command `SHOW-USER-STATUS, INFORMATION=*JOB`.

The BS2000 subsystem TANGRAM

BS2000 users can use the performance-enhancing subsystem TANGRAM (Task and Group Affinity Management) for multiprocessors (see the manual "Introduction to System Administration"). No special UDS/SQL setting is required by the database administrator for this purpose; UDS/SQL automatically reports all server tasks in a task group to TANGRAM.

TANGRAM can improve overall performance especially in cases involving UDS/SQL configurations for which the load parameter PP SERVERTASK must be set to a medium or high value, either due to strongly fluctuating loads or due to the load parameter `PP IO=SYNC` or in cases when no prioritization is possible due to other important applications.

By contrast, no significant improvement can be expected for UDS/SQL configurations that have been optimally set by other methods (fixed BS2000 priorities, UDS/SQL load parameters). TANGRAM can, however, also be used in these cases to optimize other multitask systems (e.g. openUTM) without damaging side-effects on the UDS/SQL configuration.

8.3.3 Optimized task communication

The event-driven task communication for transporting requests between the user and server task is optimized internally by UDS/SQL. No direct measures are usually required by the database administrator to control this communication.

The settings in BS2000 and the load parameters do, however, affect the communication indirectly to some extent. Consequently, to control this effect, the task communication is described below with an explanation of which monitoring counters can be used for analysis.

Description of the task communication

The transfer of a request occurs via a common memory pool called the communication pool (CUP), which is created by UDS/SQL in accordance with the specifications for the load parameter PP CUPSIZE.

The task communication is used to wait and wake up the server tasks and user tasks and is implemented by means of the BS2000 eventing mechanism.

The following diagram illustrates the task communication for each request.

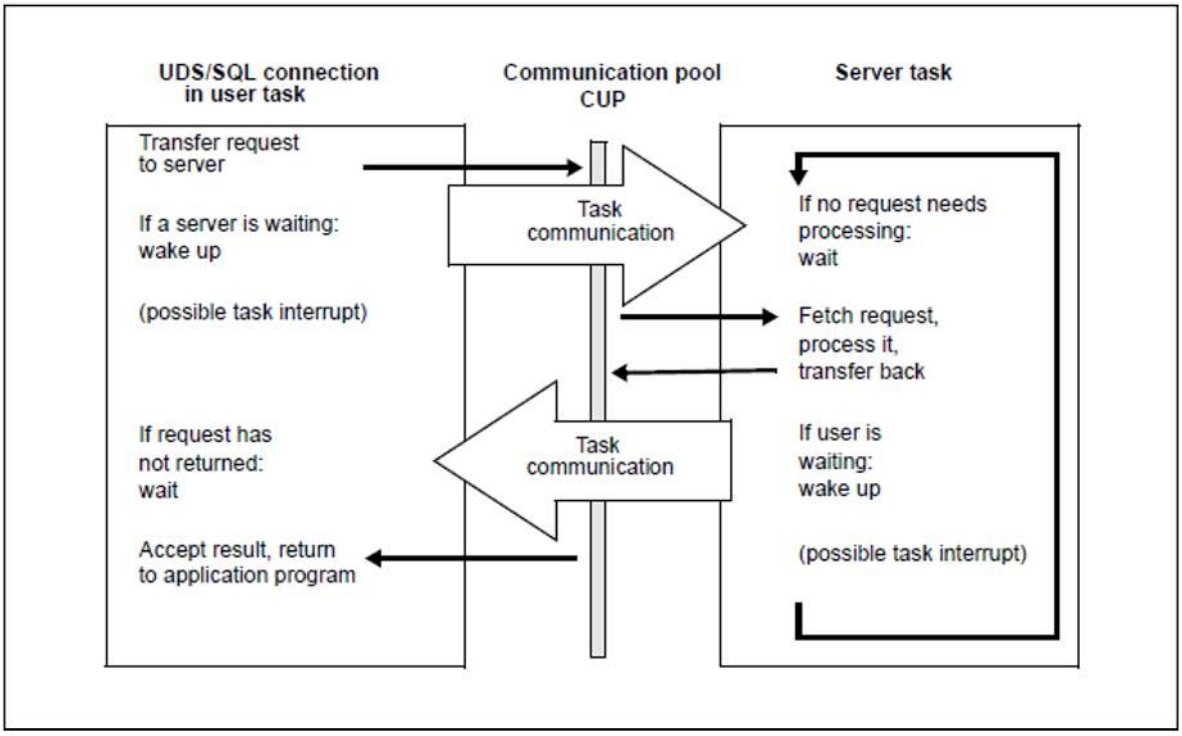


Figure 9: Task communication for each request

The objective of optimizing task communication is to avoid unnecessary wait and wake-up requests to BS2000 and to thus reduce CPU consumption. In this case, a server task is awakened to process a DML only if it has already been activated (“forward optimization”), and when the result is sent back to the user program, that program is awakened only if it has already passed a wait request to BS2000 (“backward optimization”). Both these optimization measures make use of BS2000 scheduling facilities.

8.3.4 Interpreting communication counters of the UDS/SQL monitor

One of the prerequisites for a successful forward optimization of the communication is a constant load of requests for the server tasks, i.e. there should always be enough requests present in the request pool for the server tasks to keep the tasks constantly occupied. This eliminates the need to wake up the server tasks.

Similarly, it is also possible to optimize the return to the user task. If the user task was interrupted due to BS2000 scheduling on transferring the request to the server task and is therefore not waiting, the need to wake up the user task is eliminated, so this step can be dropped.

The extent to which the optimizations work can be observed with the following monitoring counters in the COUNTER mask:

- The “DML CALLS” counter counts the total number of requests passed to the server tasks.
- The “ITC US -> ST” counter contains the number of executed wake-up signals sent from the UDS/SQL connection to the server tasks.
- The “ITC ST -> US” counter contains the number of executed wake-up signals sent from the server tasks to the UDS/SQL connection.

Detailed descriptions of these counters can be found in section “Description of UDS/SQL monitor output to a file”.

If both these two counters “ITC US -> ST” and “ITC ST -> US” are almost identical to the counter “DML CALLS”, no effective optimization in task communication could be achieved. The possible causes for this may be:

- a very low load
- a high number of server tasks (load parameter PP SERVERTASK)
- incorrectly set BS2000 priorities

If the “ITC US -> ST” counter is almost equal to the “DML CALLS” request counter of the same monitor mask, this means that the server task had to wait frequently, since they would otherwise not need to be awakened. This indicates that a user task and a server task was switched frequently on the processor, with only one request being processed in each case. In other words, the forward optimization had no effect.

If the “ITC ST -> US” counter is almost the same as the “DML CALLS” request counter of the same monitor mask, this means that the user tasks were frequently already in a waiting state when the DML request had finished processing in the server tasks. In other words, the backward optimization had no effect.

Both counters taken together could, however, never drop below the number of “DML CALLS”, since at least one task communication partner must always wait for each request.

When interpreting counters, bear in mind that the throughput of transactions is the decisive criterion for optimization. The communication counters cannot reflect all the results achieved from BS2000 scheduling; an improvement in throughput may occasionally be accompanied by a marginal deterioration in the communication optimization.

8.3.5 Example of a peak-load configuration

The following example explains the effects of the various parameters discussed earlier. It is based on the assumption that a UDS/SQL configuration, for which certain system services are to be guaranteed, is being operated on a system at peak load.

The ideal system behavior from a task communication viewpoint is to allow a few requests to be collected and then be processed by the server tasks without interruption. When these requests are being processed, further requests may also be received from other processors if there are more processors present than server tasks.

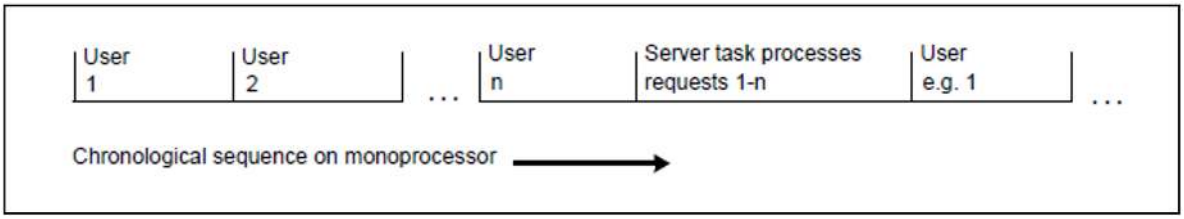
This effect can be achieved by correctly setting the BS2000 priority and by making use of a BS2000 scheduling principle. In this case, a task with a higher priority that is waiting for a processor to be allocated will not always displace an executing task with a lower priority. The displacement is to occur only if an extreme difference in priority is involved.

If the user tasks have a marginally higher priority than the server tasks, many user tasks will initially place their requests in the request pool, although a server task has already been awakened by the first user task. It is only when all user tasks have been queued for processing that the server task is allocated the processor.

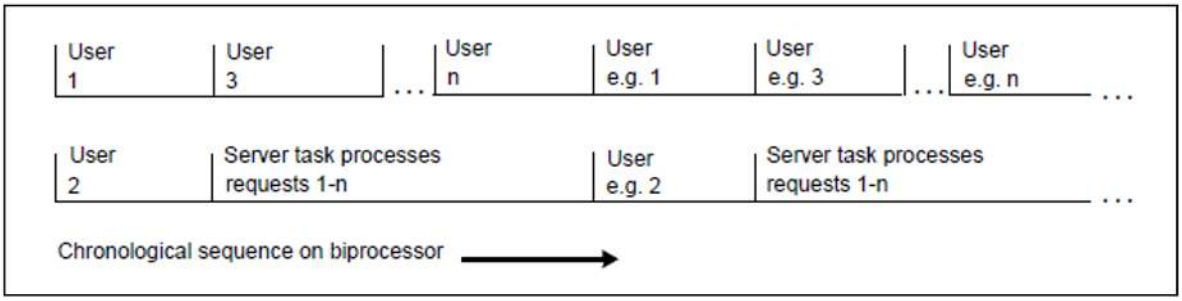
The server task is now executed on the processor and handles all pending requests in succession, even though the first user task has already been awakened following the first request.

This effect can be achieved permanently only if fixed priorities are used both for the server tasks and for the user tasks (in range 127 to 60). If variable priorities (in range 255 to 128) are used, the internal priority values will be changed dynamically by the operating system, which means that the real-time behavior of the system can no longer be predicted accordingly.

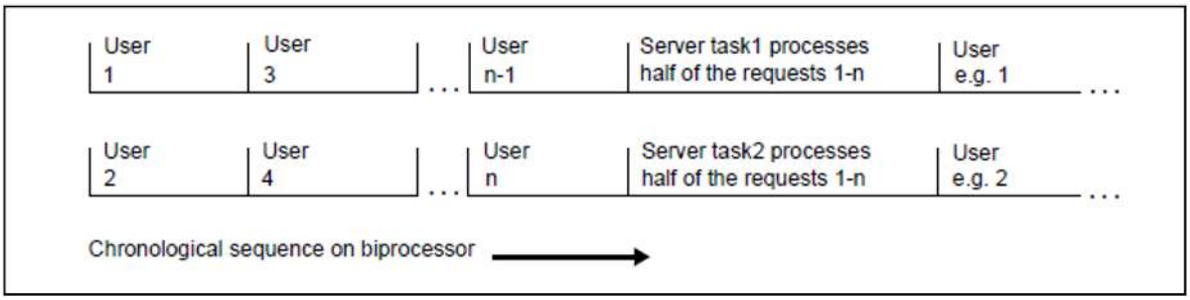
If the BS2000 priority of the master task and thus the server tasks is set to 120 (see RUN-PRIO for ENTER jobs in chapter "Starting DBH") and that of the user tasks is set to 119, the following favorable situation will arise on a monoprocessor with a load parameter setting of `PP SERVERTASK=1`:



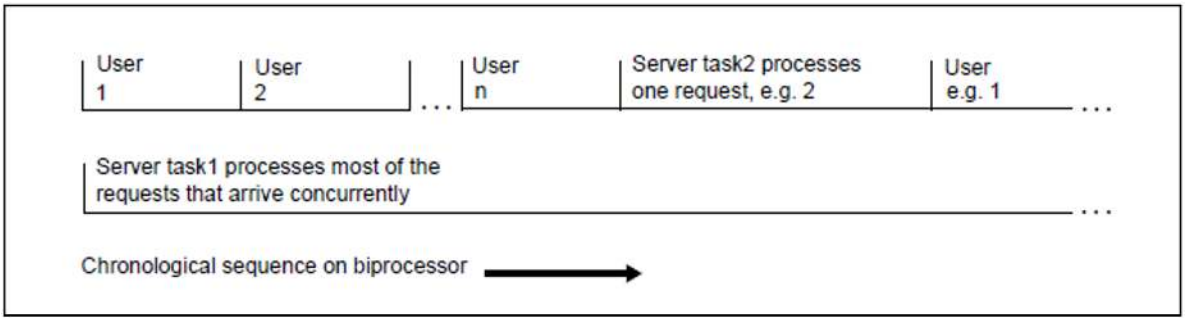
The following favorable situation occurs on a biprocessor with the load parameter setting `PP SERVERTASK=1` if the CPU share of the user programs is greater than that of the server tasks:



The following favorable situation occurs with *n* users on a biprocessor with the load parameter setting PP SERVERTASK=2:



If the situation with only one server task when the CPU share of the server tasks is greater than that of the user tasks is to be duplicated on a biprocessor with two server tasks, the load parameter `PP SCHEDULING=ASYMMETRIC` must be set. In this case, one server task will attempt to continuously process requests, while the other server task will only process one request.



The situation on a quad-processor system is analogous to that of the biprocessor with more than one server task.

The effect of the load parameter `PP SCHEDULING=ASYMMETRIC` can thus be summarized as follows: Since a server task always executes one request, all other server tasks on the parallel processors are one for a correspondingly longer period.

Other tasks with a lower BS2000 priority on the same system take up the processor time if a typical short-term reduction in the load occurs or if the database configuration requires only a part of the CPU time.

Other tasks with a higher BS2000 priority such as system tasks, for example, have no significant effect on the processing flows described above, unless such tasks occupy a processor permanently.

9 Using BS2000 functionality

This section describes how BS2000 functionality can be used in combination with UDS/SQL.

9.1 Using pubsets in UDS/SQL

The pubset management of BS2000 (see the “Introduction to System Administration”) offers you the option of determining the location at which files are stored on public disks. Both single-feature pubsets (SF pubsets) and system-managed pubsets (SM pubsets) can be used here.

The database administrator can exploit several **advantages** when utilizing the pubset administration facilities in UDS/SQL. It is possible to

- use the `CREATE-FILE` command to set up UDS/SQL files and thereby locate them on particular public volume sets
- use physical allocation to place files specifically on volume sets or individual volumes. To do this the BD2000 system administrator must make the SF pubset available with the corresponding attribute or grant the user ID the right to perform physical allocation on a pubset-specific basis. For details, please refer to the "Introduction to System Administration" and "Introductory Guide to DMS" manuals.
- use public volume sets, volume sets or volumes for database files exclusively to ensure that the disk space required for dynamic extension or online extension is also actually available.
- limit the consequences of a failure by positioning UDS/SQL files on a number of different public volume sets, so that, in the event of the failure of one public volume set, files on other public volume sets are still usable
- use state-of-the-art hardware technology (clones, snaps) by storing database files and RLOG files in a public volume set in order to be able to make database copies at any time during ongoing operation. For security reasons the RLOG file must, however, always be maintained in duplicate on a different volume set in these cases.
- restrict the UDS/SQL pubset environment (see section “Pubset declaration job variable”) to prevent the ambiguity of file names from having a disturbing influence on UDS/SQL operation.
- place the original files and their respective backup copies (`.SAVE`) on different public volume sets, in keeping with the principles of enhanced recovery
- achieve enhanced performance by expedient distribution of the UDS/SQL files over different public volume sets.

The UDS/SQL database administrator must take note of the following:

UDS/SQL does not treat existing database files and system files in the same way as new files.
Database files are: realms, ALOG file, HASHLIB and COSSD.
System files are: SLF, temporary realms, RLOG files and DB status file.

Existing database and system files

For such files UDS/SQL requires there to be no second file of the same name in any of the relevant public volume sets under their user ID in the local processor.
The public volume sets which are relevant for operating UDS/SQL can be specified for a program run by means of the pubset declaration (see section “Pubset declaration job variable”). If nothing is specified, the requirement for uniqueness applies for all public volume sets which are available locally.
Owing to this requirement for unique names, UDS/SQL itself is capable of ascertaining the catalog ID :*catid*: of any existing database or system file and then using it internally. Thus for such files it is not essential to specify the catalog ID explicitly at the user interface, and this is in fact partly forbidden by UDS/SQL so as to avoid discrepancies between the :*catid*: specified for a file and its actual :*catid*:.

In this way the database administrator can define the location of the major UDS/SQL files flexibly without having to inform UDS/SQL explicitly of the location of the individual files.

Existing files of other types

For all other files, such as load parameter files, DISTABLE files, UDS/SQL ENTER files, etc., the catalog ID can generally be specified together with the user ID when assignment takes place. However, a check is made to ensure that the scratch files of the utility routines (SCRTCH and SORTWK files) are unique in accordance with the pubset declaration. Consequently any catalog ID which is specified can be ignored.
Existing files which the database administrator has not assigned specifically must otherwise be under the default catalog ID (see the ADD-USER command in the commands manuals for "BS2000 OSD/BC").

As a rule, specifying the catalog ID is always permitted:

- when setting up files with the `CREATE-FILE` command,
- when assigning files for UDS/SQL with the `ADD-FILE-LINK` command and `LINK-NAME and`
- when declaring the configuration name with the `SET-FILE-LINK` command and `LINK-NAME.`

Files to be set up by UDS/SQL

Unless they have to be set up explicitly by the user, UDS/SQL creates any missing files under the **default catalog ID** of the database or configuration user ID (see the "BS2000 OSD/BC" command manuals, ADD-USER command).

Exception

RLOG file:

> Catalog ID defined with `PP LOG`, `PP LOG-2`, `PP RESERVE`, `MODIFY LOG`, `MODIFY-LOG-2` or `MODIFY RESERVE.`

ALOG file:

> Catalog ID defined with BMEND statement `START-LOG.`

If the default catalog ID is to be changed in the user catalog, note that all files that are not searched in the available pubsets ("files of other types“) must also be transferred from the old default catalog identifier to the new so that they, too, can be found by UDS/SQL (see the MODIFY-USER-ATTRIBUTES command in the commands manuals for "BS2000 OSD/BC").

UDS/SQL accesses locally accessible public volume sets only, even if multiprocessor systems (MSCFs) and remote file access (RFA) are used.

No database or system files are opened by UDS/SQL until an FSTAT macro has been issued (`SHOW-FILE-ATTRIBUTES` command) for all public voume sets assigned in the currently valid UDS/SQL pubset declaration. Processing does not wait until public volume sets that are in QUIET state exit this state again. As a result, files located in these public volume sets are not included in subsequent processing even if they are included in the UDS /SQL pubset declaration.

The greater the number of public volume sets occupied by database and system files belonging to a given database, the greater the internal administration overhead required. Consequently, to avoid a degradation in performance, only the required number of public volume sets should be included in the UDS/SQL pubset declaration.

Restrictions on the assignment of files with respect to the catalog ID are possible with statements of the utility routines. These are described in the descriptions of the interfaces of the individual utility routines in the related manual.

The following table indicates the user/DBH interfaces at which specifying :*catid*: is and is not permitted:

User interface	**Function**
`/SET-FILE-LINK LINK-NAME=DATABASE,` `  FILE-NAME=[:`*`catid`*`:][$`*`admuserid`*`.]`*`configuration-name`*	Defines the configuration name
`/ADD-FILE-LINK LINK-NAME=DATABASE,` `  FILE-NAME=[:`*`catid`*`:][$`*`dbuserid`*`.]`*`dbname`*`[.DBDIR]`[1]	Defines the database for the utility run
`PP DBNAME=[$`*`dbuserid`*`.]`*`dbname`*`[.`*`copy-name`*`],...` `ADD DB=[$`*`dbuserid`*`.]`*`dbname`*`[.`*`copy-name`*`],...`	Specifies the name and optionally the database user ID
`PP DISTABLE=[:`*`catid`*`:][$`*`userid`*`.]`*`file-name`* `ADD DIS,FILE=[:`*`catid`*`:][$`*`userid`*`.]`*`file-name`* `SAVE DIS,FILE=[:`*`catid`*`:][$userid.]`*`file-name`*	Specifies the name of a DISTABLE file
`/ADD-FILE-LINK LINK-NAME=PPFILE,` `  FILE-NAME=[:`*`catid`*`:][$`*`userid`*`.]`*`file-name`* `/ASSIGN-SYSDTA TO=[:`*`catid`*`:][$`*`userid`*`.]`*`file-name`*	Assigns the load parameter file

Table 20: Overview of options for the catalog identifier at UDS/SQL user interfaces

[1] Specifying :*catid*: is permitted in this case so that an existing DBDIR can be utilized for the ADD-FILE-LINK command.

Key:

userid user identification

dbuserid database identification

admuserid configuration identification

9.2 Using job variables in UDS/SQL

The following job variable types are supported by UDS/SQL:

Type	Name	Meaning	See
Pubset declaration	LINK-NAME= UDSPS01	Defines the pubsets which can be used by the DBH and the utility routines	"Pubset declaration job variable"
Session	JV-NAME= UDSSQL.DBH.*session*	Information on the status of the DBH session for automatic control of applications and administration	"Session job variable"
Database	JV-NAME= UDSSQL.DB. *database-name*	Information on the status of a database for automatic control of applications and administration	"Database job variable"
BMEND	LINK-NAME= JVBMEND	Output data of BMEND for automatic control of backup and recovery	"BMEND job variable" and the "Recovery, Information and Reorganization" manual

Table 21: Job variables used by UDS/SQL

In order to work with job variables, the “Job Variables (JV)” subsystem must be installed. If this subsystem is not available, generally no job variables are supplied and no corresponding message is issued.

9.2.1 Pubset declaration job variable

The DBH and utility routines generally check whether the files used - in particular the database files - are unique in all attached pubsets. You can use a UDS/SQL pubset declaration to restrict this check to selected pubsets.

Restricting the requirement that file names should be unique provides you with greater flexibility when organizing multiple UDS/SQL applications on one computer. Repercussions on the operation of UDS/SQL applications caused by a change to the storage organization - for example in the active use of state-of-the-art storage technologies based on clones and snaps - can be avoided.

You can optionally provide the UDS/SQL pubset declaration in a pubset declaration job variable which the DBH and the utility routines access via the predefined job variable link name UDSPS01.

If no UDS/SQL pubset declaration is available when the program starts, existing database files, system files and work files of the utility routines are searched for in the file catalogs of each locally-available pubset in which the user ID of the executing routine is entered as having access permission. This standard behavior corresponds to a UDS /SQL pubset declaration "*". Also if the "job variables" product is not available, UDS/SQL behaves as if a UDS/SQL pubset declaration "*" were present.

> ¡ If when the DBH is initialized or when a utility routine is started a faulty UDS/SQL pubset declaration is assigned or the job variable assignment is faulty, the session or utility routine is terminated.

Syntax of the UDS/SQL pubset declaration

The UDS/SQL pubset declaration in the pubset declaration job variable consists of a sequence of catid groups, each of which comprises an FSTAT-compliant catalog specification and separated from each other by one or more blanks.

The syntax of the UDS/SQL pubset declaration is therefore as follows:

catid-group[catid-group]...
Specifies one or more catid groups (1- to 4-character catalog IDs without ":") which may contain the wildcards listed in table 22 below:

*	Replaces any string, even an empty one. An * in the first position must be duplicated if the * is followed by further characters and the string entered does not contain at least one more wildcard.
/	Replaces precisely one arbitrary character.
<s_x:s_y>	Replaces a string for which the following applies: • It is at least as long as the shortest string (s_x or s_y) • It is no longer than the longest string (s_x or s_y) • In the alphabetic sort it is between s_x and s_y; digits are sorted afterletters (A...Z, 0...9) • s_x may also be the empty string which is in the first position in thealphabetic sort • s_y may also be the empty string which in this position stands for thestring with the highest possible coding (contains only the characters X'FF')

<s_1,...>	Replaces all strings which one of the character combinations specified with s matches. s can also be the empty string. Each s string can also be the range specification "s_x:s_y".
-s	Replaces all strings which do not match the specified string s. The minus sign may only be used at the beginning of the string. **This specification cannot be combined with other specifications in a UDS /SQL pubset declaration.**

Table 22: Wildcards for catalog IDs in the UDS/SQL pubset declaration

Up to 100 catid groups may be specified.

A catid group may be up to 26 characters long.

Lowercase letters are treated like the corresponding uppercase letters.

Catalog IDs which do not exist or are not available may be specified.

Multiple specification of catalog IDs is also possible.

Examples

Specification	**Stands for the following catalog IDs**
A001	A001
X/C	XAC, XBC, XCC, ..., XZC, X0C, ..., X9C (All 3-character catalog IDs which begin with X and end with C)
5*	5, 5A, ..., 59, 5AA, ..., 599, 5AAA, 5999 (All 1- to 4-character catalog IDs which begin with 5)
<C015:C025>	C015, C016, ..., C024, C025 (All 4-character catalog IDs in the range from C015 to C025)
<BE:DC>	BE, BF, BG, ..., B9, CA, ..., C9, DA, ..., DC (All 2-character catalog IDs in the range from BE to DC)
<D015:D025,F015:F045>	D015, D016, ..., D024, D025, F015, F016, ..., F044, F045 (List of ranges: all 4-character catalog IDs from D015 to D025 and from F015 to F045)
<A:D;BE:DC>	A, B, C, D, BE, BF, BG, ..., B9, CA, ..., C9, DA, ..., DC (List of ranges: all 1-character catalog IDs from A to D and all 2-character catalog IDs from BE to DC)
-5*	All (1- to 4-character) catalog IDs which **do not** begin with 5 (exclusion condition).

Table 23: Examples of catalog IDs in the UDS/SQL pubset declaration

Command string for defining and assigning a UDS/SQL pubset declaration:

```
/CREATE-JV JV-NAME=UDS-PUB-DECL
/MODIFY-JV JV-CONTENTS=UDS-PUB-DECL, -
/ SET-VALUE='A001 B001 <C015:C025> <D015:D025,F015:F045> 5*'
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
```

Command string for defining and assigning a UDS/SQL pubset declaration with exclusion condition:

```
/CREATE-JV JV-NAME=UDS-PUB-DECL
/MODIFY-JV JV-CONTENTS=UDS-PUB-DECL,SET-VALUE='-5*'
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
```

The Default Public Volume Set of the execution user ID is always implicitly taken into account by UDS/SQL and consequently does not need to be included in the UDS/SQL pubset declaration. It is not possible to exclude the Default Public Volume Set of the execution user ID from being used.

Assignment of a UDS/SQL pubset declaration which only contains blanks is permissible; only the Default Public Volume Set of the execution user ID is taken into account.

When files on private disk are used, all the pubsets whose catalogs manage files on private disk which are required for operation must be contained in the UDS/SQL pubset declaration.

Database files (realms, ALOG files, HASHLIB, COSSD), system files (SLF, temporary realms, RLOG files, DB status files) and the work files of the utility routines are included in the uniqueness check.

The DBH's load parameter files, the library files for the UDS/SQL program sections and data modules which need to be loaded or loaded dynamically (SSITAB, PLEX) and the distribution tables for UDS-D are **not** included in the uniqueness check.

In the case of load parameter `PP DISTABLE`, `DAL &ADD DISTRIBUTION` and `DAL &SAVE DISTRIBUTION` you can specify the catalog ID for the file concerned without there being any repercussions on the UDS/SQL pubset declaration.

The UDS/SQL pubset declaration is checked by the DBH or the utility routine concerned when initialization takes place. The DAL NEW PUBSETS also causes the DBH to check a newly allocated UDS/SQL pubset declaration. Faulty assignments result in the session or utility routine aborting when it starts. When the UDS/SQL pubset declaration is modified incorrectly using DAL, it is rejected and the existing UDS/SQL pubset declaration is used.

Faulty assignments can be caused by the following:

- A non-existent or inaccessible job variable is assigned
  (DBH message UDS0752, utility routine messages 0048 and 0049).
- The catid groups were specified with syntax errors
  (DBH message UDS0748, utility routine message 0045).
- An error was detected in the syntax check of a catid group
  (DBH message UDS0749 with insert CMDWCC).
- Using a catid group in the catalog access during the syntax check leads to an error (DBH message `UDS0749` with Insert `FSTAT`, utility routine message 0055).

The UDS/SQL pubset declaration must be compatible with the volume allocations for RLOG and ALOG files. The DBH checks this compatibility. Actions which lead to incompatible volume allocation are rejected.

9.2.1.1 Use of the UDS/SQL pubset declaration in the DBH

While the DBH is being initialized, any existing UDS/SQL pubset declaration is checked. The settings of an error-free UDS/SQL pubset declaration are recorded for further use.

If an error occurs in processing an allocated pubset declaration job variable while the DBH is being initialized, the session is terminated as the job variable can clearly not be used in the way that was intended.

It is recommended - especially in the case of a DBH which is running in the background - that you should assign a pubset declaration job variable in the start procedure because this cannot be done later from another task. If you do not want to restrict the pubsets which are to be taken into account, you can enter the default setting "*" in the pubset declaration job variable.

As a result you can modify the UDS/SQL pubset declaration for an active Independent DBH dynamically: enter the required UDS/SQL pubset declaration in the assigned pubset declaration job variable and then allocate the updated UDS/SQL pubset declaration using `DAL NEW PUBSETS.`

When defining a UDS/SQL pubset declaration, ensure that all the pubsets required for the application are entered so that no error occurs during operation. This applies in particular for the specifications regarding the location of the logging file made with load parameters (`LOG, LOG-2` or `RESERVE`) or DAL commands (`LOG, LOG-2` or `RESERVE`).

The effects of inconsistent specifications here are cushioned by means of the following measures:

- The DBH is not started if one of the RLOG logging specifications `PP LOG, PP LOG-2` or `PP RESERVE` is outside the pubset space of the current UDS/SQL pubset declaration.
- A modification of the RLOG logging specifications using `DAL MODIFY LOG, MODIFY LOG-2`, or `MODIFY RESERVE` is rejected if the new specification is outside the pubset space of the current UDS/SQL pubset declaration or outside the pubset space of a new UDS/SQL pubset declaration which has already been specified using `DAL NEW PUBSETS` but has not yet been activated by `PERFORM`.
- A database cannot be attached if one of the ALOG logging specifications which were defined using the `DEFAULT-SUPPORT` or `RESERVE-SUPPORT` parameter of the BMEND statement `START-LOG` is outside the pubset space of the current UDS/SQL pubset declaration.
- Modification of the current UDS/SQL pubset declaration by means of `DAL NEW PUBSETS` is rejected if the new UDS/SQL pubset declaration does not include the pubsets which are predefined for `LOG, LOG-2, RESERVE` of RLOG logging and for `DEFAULT-SUPPORT` and `RESERVE-SUPPORT` of ALOG logging for each attached database.

The settings of a faulty UDS/SQL pubset declaration are logged. You can display the current UDS/SQL pubset declaration using `DAL DISPLAY PUBSETS.`

9.2.1.2 Use of the UDS/SQL pubset declaration in the utility routines

When a utility routine starts, any UDS/SQL pubset declaration which exists is checked. The settings of an error-free UDS/SQL pubset declaration are recorded for further use, but they are not logged. A faulty job variable assignment or the assignment of a faulty UDS/SQL pubset declaration when the utility routine starts results in the utility routine aborting.

When DEFAULT-SUPPORT and RESERVE-SUPPORT of ALOG logging are specified with the BMEND statement START-LOG, no check against the assigned UDS/SQL pubset declaration takes place as the specification may be defined for another application environment.

9.2.1.3 Other uses of the UDS/SQL pubset declaration

A UDS/SQL pubset declaration does not apply for the COBOL compiler which reads the COSSD file when a UDS /SQL application program is compiled using COBOL-DML. The COSSD file can be assigned explicitly to the COBOL compiler, together with the catalog ID, using the `ADD-FILE-LINK` command with `LINK-NAME=UDSCOSSD` (see the "Application Programming" manual, section "Special features of COBOL-DML").

There is no special treatment for files from an alias catalog (ACS). Above all you should not permit any aliases with catalog and/or user ID (`COMPLETE-ALIAS-NAMES=*ALLOWED`) as this would conflict with the intended effects of the UDS/SQL pubset declaration (see section “ACS (Alias Catalog Service)”).

9.2.2 Session job variable

The DBH supports a session job variable that helps to automate administration. This job variable can be used to control user jobs and programs.

The following job variable is supplied by UDS/SQL:

UDSSQL.DBH.*session*

session

is the variable part of the job variable name. The first eight characters of the full *configuration-name* is used as the default value. If desired, the database administrator can also define some other name for *session* before starting the DBH in the master task or in the linked-in task by using the following command:
`/SET-FILE-LINK LINK-NAME=UDSDBHJV,FILE-NAME=session`

If the job variable UDSSQL.DBH.*session* does not exist, it is created with the following attributes:

```
ACCESS=WRITE
SHARE=YES
BASACL_NONE (standard access control)
```

If the job variable is to be used with other properties, you must create it with the required properties before the DBH starts or before the utility routine executes. When using job variables, access control can be implemented using standard resources (e.g. Basic-ACL, Guards).

Before a job variable is used in a foreign user ID, you must either create it yourself or take appropriate measures (e. g. co-ownership in a foreign user ID) to ensure that it can be created dynamically.

> ¡ Please also take note of “Information on using the job variables”.

The job variable UDSSQL.DBH.*session* is structured as follows:

Column	Contents	Meaning
1-30	UDSDBH_STARTING / UDSDBH_ACTIVE / UDSDBH_CLOSE_INITIATED / UDSDBH_NORMAL_END / UDSDBH_ABNORMAL_END	Status (1)
		DBH start (2)
31-40	*yyyy-mm-dd*	Date
41-48	*hh*:*mm*:*ss*	Time
		DBH end (3)
49-58	*yyyy-mm-dd*	Date
59-66	*hh*:*mm*:*ss*	Time

67-68	01	Version ID of the layout of the session job variable
69-76	*session-name*	Configuration name of the DBH session
77-84	*host*	Host of the DBH session
85-92	*nnnnnnnn* / 'BLANK'	Session section number (4)
93		Status of the DBH operational interruption:
	B /	– Current operational interruption
	'BLANK'	– Currently no operational interruption
94-96		Type of current or last DBH operational interruption (5)
	DAL /	– Because of DALs (PERFORM)
	DBH /	– Because of internal system activities
	'BLANK'	– No operational interruption has occurred as yet
97		Attaching database or realm in current or last operational interruption (5)
	A /	– Yes
	– /	– No
	'BLANK'	– No operational interruption has occurred as yet
98		Detaching database or realm in current or last operational interruption (5)
	D /	– Yes
	– /	– No
	'BLANK'	– No operational interruption has occurred as yet
99		Activity with respect to ALOG files in current or last operational interruption (5)
	A /	– Yes
	– /	– No
	'BLANK'	– No operational interruption has occurred as yet
100		Activity with respect to RLOG files in current or last operational interruption (5)
	R /	– Yes

	– /	– No
	'BLANK'	– No operational interruption has occurred as yet
101-112	'BLANK'	Reserved
		Start of the preparatory phase for the current or last operational interruption (5)
113-122	*yyyy-mm-dd*	Date
123-130	*hh*:*mm*:*ss*	Time
131-138	*hh*:*mm*:*ss*	Start of the Implementation phase for the current or last operational interruption (5) Time
139-146	*hh*:*mm*:*ss*	End of the last operational interruption (5) Time
147-162	'BLANK'	Reserved
		Last change to the job variable
163-172	*yyyy-mm-dd*	Date
173-180	*hh*:*mm*:*ss*	Time

Table 24: Structure of the session job variable for UDS/SQL

Comments

(1) *Status*

Content	Set when?
UDSDBH_STARTING	When the DBH starts
UDSDBH_ACTIVE	After successful initialization (e.g. System Ready message of the independent DBH)
UDSDBH_CLOSE_INITIATED	When no further requests are permitted in the DBH because the DBH end has been initiated by the UDS/SQL administration or for internal reasons by the DBH
UDSDBH_NORMAL_END / UDSDBH_ABNORMAL_END	When the DBH ends

(2) *DBH start* is initialized with 0 when the DBH starts and is filled with the current time following successful initialization (e.g. System Ready message of the independent DBH).

(3) *DBH end* is initialized with 0 when the DBH starts and is filled with the current time when the DBH ends.

(4) *Session section number* is initialized with blanks when the DBH starts and during initialization is filled with a value which unambiguously identifies the session section and remains unchanged up to the end of the session section.

The session section number enables the assignment of database job variables to a current session to be checked: only if the session section numbers in the database job variables and in the session job variables match are the contents of the database job variables valid for the current session.

(5) The displays for and times of the various phases of an operational interruption always relate to the current or last operational interruption displayed in *Status of the DBH operational interruption* (byte 93).

Internal operational interruptions in the DBH’s start or termination phase are not displayed.

9.2.3 Database job variable

The DBH and the utility routines DDL compiler, SSL compiler, BGSIA, BGSSIA, BPRIVACY, BMEND, BREORG, BCHANGE, BRENAME and BALTER maintain a database job variable to enhance automatic administration. You can use this job variable to control user requests and programs.

The name of the database job variable has the following format:

UDSSQL.DB.*databasename*[.*copyname*][.*no*]

databasename

Name of the database (up to 17 characters)

copyname

Copy name of a database attached in the DBH in SHARED-RETRIEVAL mode

no

Single-digit number (1..9) of the database in the case of databases which are used in various sessions in parallel in SHARED-RETRIEVAL mode.
Utility routines only use a job variable without *no*.

The database job variable is always located in the user ID and in the pubset of the DBDIR of the corresponding database.

If no database job variable exists at initialization time, it is created with the following properties:

```
ACCESS=*WRITE
USER-ACCESS=*ALL-USERS
BASIC-ACL=*NONE (simple access control list)
```

If the job variable is to be used with other properties, you must create it with the required properties before the DBH starts or before the utility routine executes. When using job variables, access control can be implemented using standard resources (e.g. Basic-ACL, Guards).

Before a job variable is used in a foreign user ID, you must either create it yourself or take appropriate measures (e. g. co-ownership in a foreign user ID) to ensure that it can be created dynamically.

Please also take note of “Information on using the job variables”.

Special conditions in the case of database in SHARED-RETRIEVAL mode

When a database is opened in SHARED-RETRIEVAL mode, the DBH first checks whether suitable job variables with numbering already exist. If one of these job variables contains the current name of the configuration, it is used, otherwise the DBH selects a job variable name which has not yet been used, creates the corresponding job variable, and enters the current configuration name. Job variables which are empty or have not yet had configuration names assigned are not used because when databases are used in parallel in multiple DBH sessions it must be ensured that a different database job variable is used by each DBH session. A corresponding job variable with no number is included in this process and reserved with priority in the event of free selection.

Database job variables in SHARED-RETRIEVAL mode are always initialized fully when the database is attached.

In the case of genuinely parallel SHARED-RETRIEVAL mode it is advisable to configure the database job variables before using them and to supply the configuration name of the DBH session (columns 61-68).

No special rules need to be observed when numbering the job variables. If all of the 10 possible job variables are in use and are therefore assigned to other configurations, no job variable can be created for the database concerned. A message to this effect is issued by the DBH.

Structure of the database job variable

The database job variable UDSSQL.DB.*databasename*.[.*copyname*][.*no*] is assigned values as follows:

Column	Contents	Meaning
1-2	01	Version identifier of the session job variable's layout
3-19	*ccccccccccccccccc*	Database name (1)
20-26	*cccccc* / 'BLANK'	Copy name (2)
27-32	*cccccc*	DB layout version
33	C / I / 'BLANK'	Consistency (3)
34-40	'BLANK'	Reserved
41-50		Processing status (4):
	UPDATE / RETR / WARMSTART /	– Attach mode for the DBH
	OPEN /	– Attach mode for a utility routine
	DROP / CLOSE / ERROR	– Cause of the last termination in the case of the DBH or a utility routine
51	'BLANK' / A	Activation of ALOG (5)
52	'BLANK' / O	Identifier for online backup capability (6)
53-60		DBH or utility routine (7) Current or last user of the database
	DBH /	– Independent DBH
	LKIN-DBH /	– Linked-in DBH
	UTIL-DBH /	– Utility routine with linked-in DBH (DDL/SSL compiler, BGSIA, BGSSIA, BPRIVACY)
	utility	– Name of a modifying utility routine
61-68	*cccccccc* / 'BLANK'	Configuration name of the DBH session (8)
69-72	*cccc* / 'BLANK'	Default catalog of the DBH user ID (9)

73-81	$*cccccccc* / 'BLANK'	User ID of the DBH session (with $) (9)
82-89	*nnnnnnnn* / 'BLANK'	Session section number (10)
		Start of processing (11)
90-99	*yyyy-mm-dd*	Date
100-107	*hh*:*mm*:*ss*	Time
		End of processing (12)
108-117	*yyyy-mm-dd*	Date
118-125	*hh*:*mm*:*ss*	Time
		Last ALOG change (13)
126-135	*yyyy-mm-dd*	Date
136-143	*hh*:*mm*:*ss*	Time
144-152	*nnnnnnnnn* / 'BLANK'	ALOG sequence number (14)
153-162	*nnnnnnnnnn* / 'BLANK'	Size of the ALOG file (15)
163-167	nnnnn / 'BLANK'	Extents of the ALOG file (16)
168-182	'BLANK'	Reserved
		Last change job variable
183-192	*yyyy-mm-dd*	Date
193-200	*hh*:*mm*:*ss*	Time

Table 25: Structure of the database job variable for UDS/SQL

Comments

(1) *Database name* is the name of the database which is also contained in the job variable name.

(2) *Copy name* is only filled with values in the shadow database.

(3) *Consistency* is only filled with values by the DBH and shows whether the database is consistent ("C") or inconsistent ("I"). In this sense a database is inconsistent when the DBH is currently modifying the database and possibly not all changes have been written back to the database files.

However, the database can also be inconsistent if processing by the DBH was terminated abnormally. Warm-starting the database generally returns it to a consistent status.

(4) *Processing status* shows the attach mode or the cause of the last termination. Temporary access restrictions (e.g. because of DAL ACCESS) or restrictions to the operating mode (because, for example, the RLOG file cannot be used) are not displayed.

ERROR is set if the database is switched off under control while it is inconsistent or processing of the database has been terminated because of other errors. In the latter case it is possible that the database is still consistent.

However, in some cases it is possible that the DBH session is terminated abnormally without it being possible to update the processing status. The database is then generally still inconsistent and the processing status remains UPDATE. When a subsequent warm start is performed by the DBH, the processing status WARMSTART is set, and at the end of the warm start the status required for processing (e.g. UPDATE) is entered. Modifying utility routines supply the statuses OPEN, CLOSE and, if required, ERROR.

In some error situations controlled program termination is not possible for the utility routines. In such a case the ERROR status can also not be set correctly. The job variable then remains in the OPEN status even after the utility routine has terminated.

It is also possible that an error occurs in the last phase of utility routine termination after the job variable has already been supplied with the CLOSE status. In this case the job variable is, if possible, filled with the ERROR status.

(5) *Activation of ALOG* shows whether ALOGGING is activated ("A") for the database or not (blank). Current processing in the DBH can take place without ALOGGING if, for example, the database is attached in SHARED-RETRIEVAL mode.

(6) *Identifier for online backup capability*
"O" indicates that an online backup of the database can currently be created, i.e. online backup capability is enabled for the database (BMEND ENABLE-ONLINE-COPY) and the database is attached to a DBH.
In all other cases the item contains blanks.
This property should always be checked before an inconsistent copy is made in parallel to a DBH session using COPY-FILE.
The item is filled with information **only by the DBH** and is deleted when the database is detached. In particular it is **not** set by BMEND if online backup capability has just been enabled.

(7) *DBH or utility routine* is filled with information by the DBH or utility routine when the database is attached and then remains unchanged.

(8) *Configuration name of the DBH session* shows the current or the last name of the DBH configuration. The item is filled with information when the database is attached and then remains unchanged. Utility routines which do not use the linked-in DBH enter blanks. In SHARED-RETRIEVAL mode a database job variable should be created before the database is attached to a DBH session and filled with the configuration name.

(9) *Default catalog of the DBH user ID* and *User ID of the DBH session* contain the values of the relevant DBH session. These values can be used for unique access to the session job variable.

Utility routines which do not use the linked-in DBH enter blanks.

(10) *Session section number* is filled with the current value of the session when the database is attached, and the DBH deletes it with blanks when the database is detached.

Utility routines which do not use the linked-in DBH enter blanks when the database is attached.

(11) *Start of processing* is filled when the database is attached (DBH) or opened (utility routine) and then remains unchanged.

(12) *End of processing* is filled when the database is detached (DBH) or closed (utility routine). When utility routines terminate abnormally, the data provided depends on whether the job value as a whole is updated. In the UPDATE and RETR status the item contains blanks.

(13) *Last ALOG change* shows the time of the last ALOG change or (re)activation of ALOGGING. The date is retained even when ALOGGING is disabled.

(14) *ALOG sequence number* is filled with values by the DBH or utility routines in the event of a change of ALOG if ALOGGING is enabled.

(15) *Size of the ALOG file* shows the size of the used part of the ALOG file in PAM pages. This size can differ slightly from the file size as seen by the DMS. This deviation can be caused either by rounding on account of the size of the allocation unit used (3, 4 or 32 PAM pages) or because as a result of doubling the secondary allocation (cf. class-2 system parameter DMMAXSC or the MAXIMAL-ALLOCATION parameter in ADD-/MODIFY-MASTER-CATALOG-ENTRY) a current file extension is larger than the extension requested by UDS/SQL.

The item is filled with a current value **only by the DBH**. Utility routines which do not use the linked-in DBH enter blanks.

(16) *Extents of the ALOG file* shows the number of extents of the ALOG file.

The item is filled with a current value **only by the DBH**.
Utility routines responsible for database transforming (BRENAME, BCHANGE, BALTER, BREORG, BMEND) enter blanks.

Output example:

```
%01MDV29B6                  004.00C       UPDATE    A DBH     MDV29B6 IUDS$UDSDEV
024AD708EF2019-01-2512:36:10                  2019-01-2512:34:520000000010000000
192 00001              2019-01-2512:36:10
```

Where:

- ALOG sequence number = 000000001
- Size of the ALOG file = 0000000192
- Extents of the ALOG file = 00001

9.2.4 Information on using the job variables

Input

The job variables are always used by UDS/SQL in their entire length. Free areas are reserved for future extensions. Changes which are performed externally are consequently overwritten and should be avoided as they can result in undefined job variable contents.

Evaluation

In the case of automatic monitoring of individual statuses or changes to the content of an arbitrary item it must be borne in mind that changes can occur very rapidly one after the other. Under some circumstances it may therefore not be possible to intercept all the intermediary statuses.

The content of session and database job variables should not be used to check whether the processing routines are active. This information must be obtained from appropriate monitor job variables.

Malfunctions

Temporary malfunctions in supplying session or database job variables are reported once only the first time they occur.

If the DBH cannot assign a value to a job variable because a recoverable malfunction has occurred, message UDS0700 with the insert `JOBV_SYSTEM_ERROR` and return information (`SEC-RC`) is issued. The session is continued.

You can generally use `HELP-MSG MSG-ID=JVSxxxx` (`xxxx`=the last four characters of SEC-RC) to determine and possibly remove the cause of the malfunction.

When a permanent malfunction of the job variable subsystem is detected, message UDS0700 with the insert `JOBV_PERMANENT_ERROR` is issued. The session is continued.

Once a malfunction in an ongoing session has been removed, the session job variable is provided with the current value without any additional message the next time input is supplied. The same applies for the database job variable the next time the database is attached.

9.2.5 BMEND job variable

The BMEND utility routine optionally assigns values to a job value when information which is relevant to AFIM-Logging is generated (`SHOW-LOG-INFORMATION statement`) or when AFIMs are applied to a database (`UPDATE-DATABASE` statement). This allows database backup and recovery operations to be automated. Details of this are provided in the description of the BMEND utility routine in the “Recovery, Information and Reorganization" manual.

9.3 Using multiple UDS/SQL versions concurrently

Different versions of UDS/SQL can be used concurrently in the same BS2000 system. UDS/SQL uses the 7-character version identifier. Correction versions of a main version can thus also be run concurrently; this only requires the correction versions to be installed under different user IDs. This is useful, for example, if a correction version is to be installed and tested while productive operation continues with the previous version.

Multiple UDS/SQL subsystems with different (correction) versions may be loaded simultaneously.

If multiple UDS/SQL versions are installed or loaded as subsystems and none of them is specifically selected, BS2000 establishes the connection to one version by default.

With IMON-GPN you can explicitly specify in each task which UDS/SQL version is to be used by issuing the following BS2000 command in the start procedure of the master task or the application program:

```
SELECT-PRODUCT-VERSION PRODUCT-NAME=product,VERSION=version,SCOPE=*TASK
```

The version number should always be specified to include the release status and correction status (e.g. 02.9C00).

The version assignment affects versions which are entered in the Software Configuration Inventory (SCI), subsystems which are loaded concurrently or start commands of UDS/SQL which are installed concurrently.

The respective version numbers specified with SELECT-PRODUCT-VERSION do not need to be permanently assigned in each user procedure, but can be structured variably and set via a centrally kept value (e.g. for a configuration in a job variable of the associated DBH identifier or in a structured variable).

Information on which versions of UDS/SQL are available and where the product files are installed can be obtained using the following command:

```
SHOW-INSTALLATION-PATH INSTALLATION-UNIT=UDS-SQL(VERSION=*ALL)
```

With ENTER jobs you can issue the SELECT-PRODUCT-VERSION command in an ENTER file.

If IMON-GPN is used, the version set in the master task is also used in the server tasks and in the UDS-D task.

Example: Parallel use of two UDS/SQL versions

The correction version UDS-SQL V2.9C10 is to be tested while productive operation continues with UDS-SQL V2.9C00. Using job variables, it should be possible to load the desired version when the DBH is started. This involves the use of parallel subsystems and start commands.

Creating job variables

The database administrator can create job variables which are to contain the values for the respective subsystem versions. The job variable JV.SHIPPING is to receive the value `02.9C00`, and the job variable JV.TEST the value `02.9C10`. The contents of the job variables are then checked.

```
/CREATE-JV JV=JV.SHIPPING
/MODIFY-JV JV=JV.SHIPPING,SET-VALUE=C'02.9C00'
/CREATE-JV JV=JV.TEST
/MODIFY-JV JV=JV.TEST,SET-VALUE=C'02.9C10'
/SHOW-JV JV=JV.SHIPPING
%  02.9C00
/SHOW-JV JV=JV.TEST
%  02.9C10
```

Creating DBH startup procedures

In the DBH startup procedures DBHSTART.TEST and DBHSTART.SHIPPING, the corresponding job variable name is to be used for the desired version.

DBHSTART.TEST

```
/SET-LOGON-PARAMETERS
/ASSIGN-SYSDTA TO=*SYSCMD
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=TEST
/CREATE-FILE FILE-NAME=TEST.DBSTAT,SUPPRESS-ERRORS=*FILE-EXISTING
/CREATE-FILE FILE-NAME=TEST.DBSTAT.SAVE,SUPPRESS-ERRORS=*FILE-EXISTING
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.TEST ---------------------------- (1)
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=&(*VERS) ----------- (2)
/START-UDS-DBH
/ASSIGN-SYSDTA TO=*PRIMARY
/EXIT-JOB
```

DBHSTART.SHIPPING

```
/SET-LOGON-PARAMETERS
/ASSIGN-SYSDTA TO=*SYSCMD
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING
/CREATE-FILE FILE-NAME=SHIPPING.DBSTAT,SUPPRESS-ERRORS=*FILE-EXISTING
/CREATE-FILE FILE-NAME=SHIPPING.DBSTAT.SAVE,SUPPRESS-ERRORS=*FILE-EXISTING
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.SHIPPING ------------------------- (1)
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=&(*VERS) ----------- (2)
/START-UDS-DBH
/ASSIGN-SYSDTA TO=*PRIMARY
/EXIT-JOB
```

1. The desired job variable name is linked with the link name VERS.
2. For the link name VERS the contents of the defined job variable are used.

As well as the DBH startup procedures, the startup procedures for application programs must also be adjusted.

Creating startup procedures for application programs

In the startup procedures of the application programs DMLTEST.TEST and DMLTEST.SHIPPING, the corresponding job variable name is to be used for the desired version.

DMLTEST.TEST

```
/BEGIN-PROCEDURE
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=TEST
/ASSIGN-SYSDTA TO=*SYSCMD
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.TEST
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=LMS.SSITAB
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=&(*VERS)
/START-UDS-DMLTEST
DBH IND
LANG COB
DISPLAY RCODE,COND=RCODE NE C‘00000‘
PROT ON
DISPLAY RECA,L=80
SUBSCHEMA IS ADMIN
READY
SHOW SPP1
SHOW SUBS
E
SYS
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*TIME(TIMEOUT=*STD)
/ASSIGN-SYSDTA TO=*PRIMARY
/RESUME-PROGRAM
/END-PROCEDURE
```

DMLTEST.SHIPPING

```
/BEGIN-PROCEDURE
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=LMS.SSITAB
/ASSIGN-SYSDTA TO=*SYSCMD
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.SHIPPING
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=&(*VERS)
/START-UDS-DMLTEST
DBH IND
LANG COB
DISPLAY RCODE,COND=RCODE NE C‘00000‘
PROT ON
DISPLAY RECA,L=80
SUBSCHEMA IS ADMIN
READY
SHOW SPP1
SHOW SUBS
E
SYS
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*TIME(TIMEOUT=*STD)
/ASSIGN-SYSDTA TO=*PRIMARY
/RESUME-PROGRAM
/END-PROCEDURE
```

For the meanings of the procedure parameters, see the explanations of the DBH startup procedures in section "Creating DBH startup procedures".

If the test version is to be used later as a productive version, the startup procedures remain unchanged. You need only change the contents of the JV.SHIPPING job variable as follows:

```
/MODIFY-JV JV=JV.SHIPPING,SET-VALUE=C'02.9C10'
```

9.4 Using DAB caching for UDS/SQL

Software caching (DAB) is useful for database operation. For reasons of data safety, it is essential to avoid write-buffering or write/read-buffering in volatile cache media (main memory, expanded memory).

This applies both to the database files and the session and logging files. The UDS/SQL safety concept (see section “The UDS/SQL recovery concept”) assumes that if a write operation is logically concluded then it is also physically concluded and will therefore survive a system crash, for example due to a power failure.

For a detailed description of DAB caching, see the manual "DAB (BS2000)".

9.5 Using data mirroring for UDS/SQL

In BS2000, it is possible to use a variety of methods for transparent data mirroring, e.g. DRV (Dual Recording by Volume), SRDF (Symmetrix Remote Data Facility) or TimeFinder.

UDS/SQL can use this transparent mirror data. This greatly increases the error-tolerance of the corresponding files since they are present in duplicate. If one unit fails, operation can be continued from the mirror disk without interruption

However, following a system crash it is possible that the two units will have different states which must first be aligned.

If DRV is used then, for example, BS2000 supports the automatic alignment of the DRV disks following a crash. However, this process can result in very long reconstruction times since the operating system has to read and write the entire disk. The DBH and the utility programs therefore perform a suitable alignment of those database areas that have differing states. This makes a potentially more time-consuming alignment operation by the operating system unnecessary.

9.6 Using HSMS in UDS/SQL

HSMS (Hierarchical Storage Management System) can be used to make online backups of UDS/SQL database files (see section “Saving a database online”).

HSMS also permits the automatic export to alternative media of files that have not been accessed for a considerable period, see the manual "HSMS / HSMS-SV (BS2000)". If an exported file is accessed then HSMS retrieves it automatically.

In principle, UDS/SQL files can also be swapped out using HSMS.

> If a database is inconsistent as an online backup or due to a DBH crash and it is swapped out using HSMS before the warm start, the DBH may reject the warm start after the file has been swapped back in. In this case, you must use the BMEND utility routine to update the database to the point at which the crash occurred before you attempt to perform a warm start (see the manual "Recovery, Information and Reorganization", BMEND statement UPDATE-DATABASE).

9.7 Using FASTPAM in UDS/SQL

UDS/SQL uses the FASTPAM access method in the independent DBH for all files, except the following:

- Realms with a 2-Kbyte page format
- Database files located on shared or protected private disks (SPD or PPD)
- Database files located on multiple public volume sets (MPVS).

These files are still processed by UDS/SQL with the UPAM access method. The linked-in DBH only used the UPAM access method.

Access with FASTPAM is more efficient than with UPAM, since some access paths are prepared for FASTPAM and some areas are maintained in resident memory.

In order to create memory-resident areas, the BS2000 user ID will need to have FASTPAM privileges. The BS2000 command SHOW-USER-ATTRIBUTES can be used to check whether the BS2000 user ID has the required privileges. The DMS-TUNING-RESOURCES item must be set to the value *EXCLUSIVE.

FASTPAM may be used even if the BS2000 user ID does not have FASTPAM privileges, but no memory-resident areas can be maintained in that case. This also applies when the BS2000 user ID has the required FASTPAM privileges, but when all I/O areas cannot be made memory-resident (due to a lack of available main memory).

If the loading of UDS/SQL is rejected under the BS2000 user ID despite an adequate value for RESIDENT PAGES, the maximum number of resident main memory pages allowed may need to be raised accordingly by the BS2000 system administrator (using the BS2000 command `MODIFY-SYSTEM-BIAS`). The currently set maximum value (CORE) can be checked with the BS2000 command `SHOW-SYSTEM-STATUS` via the parameter `INFORMATION=*SYSTEM-PARAMETERS.`

You specify the required number of resident pages via the RESIDENT-PAGES operand when the DBH is started (`START-UDS-DBH` or `START-EXECUTABLE-PROGRAM` command). You must specify a minimum value and a maximum value.

If you specify a minimum value = 0, UDS/SQL will start up in any case, but possibly with degraded performance. If the specified minimum value is > 0, UDS/SQL is started only if the minimum requirement can be satisfied.

In order to calculate the maximum value, the sizes of the 4-Kbyte and 8-Kbyte buffer pools, in particular, must be taken into account. For example, the approximate value for a total of *n* databases would be

300 Kbyte + *n**100 Kbyte + 1,1 * (sum of the sizes for the 4-Kbyte and 8-Kbyte buffer pools)

Since the 4-Kbyte system buffer pool is at least 1.3 Mbytes in size, the minimum requirement (for an attached database) would be 1.83 Mbytes = 458 resident 4-Kbyte pages.

If the return code UDS0600 is output with the insert USER LIMIT EXCEEDED on starting the DBH, then the DBH has not been assigned enough resident pages. The reason for this can be that the maximum value specified for RESIDENT-PAGES at startup was too small, but also that the system could not make enough pages available.

More information on the FASTPAM access method can be found in the manual “Introductory Guide to DMS”.

9.8 Using the full address space in UDS/SQL

All available address space (XS) can be used in UDS/SQL.

UDS/SQL tasks of the independent DBH are always executed in addressing mode (AMODE) 31.

UDS/SQL code in user tasks (e.g. the UDS/SQL connection) and in the linked-in DBH is executed in the AMODE of the application program that issues the DML request. In the linked-in DBH, the AMODE must not be modified dynamically in an application program run for the DML requests.

User code that is directly entered during DML processing (e.g. the DSCEXT error handling routine of CALL DML) must be executable in the AMODE of the DML request.

It is generally advisable to make all application programs XS-compatible and to run them in AMODE 31. This will avoid the problem of address space restrictions when more complex and extended functionality is introduced in the applications.

9.9 Determining time via the GET-TIME subsystem

UDS/SQL uses the BS2000 subsystem GET-TIME to determine time information.

GET-TIME must be started during BS2000 system initiation in order to start a UDS/SQL configuration. The GET-TIME parameters for time zones, skipped time, and daylight savings time (i.e. summer and winter time) and their associated conversion data must be correctly supplied to ensure that the UDS/SQL configuration operates smoothly. The conversion data must extend as far back into the past as required for the UDS/SQL data repositories to be processed during the BS2000 session.

If UDS/SQL code is to be executed in the AMODE 24 addressing mode, the BS2000 system administrator must ensure that the GET-TIME subsystem is loaded below the 16-Mbyte boundary. A special variant of declaration file for the GET-TIME subsystem is used for this purpose.

If UDS/SQL is being used as a DSSM subsystem, the BS2000 system administrator must explicitly specify the dependency on GET-TIME subsystem when generating the UDS/SQL subsystem. This can be done by specifying the parameter `REFERENCED-SUBSYSTEM=GET-TIME` in the SCCM statement `SET-SUBSYSTEM-ATTRIBUTES` (see also the manual “Subsystem Management in BS2000”).

9.10 Assigning passwords to UDS/SQL files

UDS/SQL protects files that are generated automatically by the DBH (i.e. the SLF, ALOG file, RLOG file and temporary realms) by means of a password.
In some cases such a password is not used to protect the contents of the files against unauthorized access; it is used to prevent files which are employed in various sessions from being inadvertently deleted. These files, which are protected by the unchangeable password C'UDS'BLANK'', only contain database management data.
The SLF is always protected with the default password: C'UDS'BLANK''. The ALOG file and the temporary files are also protected with the default password C'UDS'BLANK'', assuming that no other password was specified with the DBH load parameter `PP CATPASS`.

As the database administrator, you can also assign passwords to UDS/SQL files. This can be done by using the BS2000 command `MODIFY-FILE-ATTRIBUTES` (see the commands manuals for "BS2000 OSD/BC"). The only exception is the RLOG file. Each RLOG file is always protected with a specific UDS/SQL-internal password.

As the RLOG files are generally deleted again by the DBH when they are no longer required, you do not need to manage the password. However, if in exceptional cases an RLOG file does need to be deleted for other reasons, you can issue the password using the RLOGPASS utility routine in the UDS-SQL-T package.

The passwords for UDS/SQL files must be made known with the load parameter `PP PASSWORD` or the DAL command `ADD PW`.

The following points should be considered when assigning passwords to UDS/SQL files:

- A read password should be assigned to all UDS/SQL files as minimum protection against the access of non-UDS /SQL systems.
- Assigning a write password is recommended only where selected non-UDS/SQL read accesses are to be granted without the files forfeiting the write protection.
- Password assignment should be economical; not every file requires its own password.
- The password C'UDS'BLANK'' and read passwords specified via PP CATPASS, PP PASSWORD or the DAL command ADD PW should not be used for other files, because UDS issues these passwords internally, so in special situations files would become accessible which users want to protect.

Special consideration must be given to the COSSD file, because it is required for COBOL compilations (COBOL85 oder COBOL2000).

If this file is to be protected, the database administrator must either

- make available to the user a compilation procedure containing the COSSD password. This procedure must itself be protected by a read and execute password, and only the execute password must be known to the user.

or

- where extreme security is required, perform compilations of COBOL-DML programs on the user's behalf.

When passwords are assigned to the file containing the DBH load parameters, care should be taken that UDS/SQL is always granted access. UDS/SQL must have write access to the following files:
the SLF, temporary files and the DB status file.
UDS/SQL must have write access to realms and ALOG files unless their databases are run in SHARED-RETRIEVAL mode.

Organizational measures

The following organizational measures are recommended when passwords are to be assigned to UDS/SQL files:

- Make use of BS2000 password encryption at system generation time (see the commands manuals for "BS2000 OSD/BC", ADD-PASSWORD command).
- Assign a logon password for the configuration user ID.
- Files containing passwords, such as CALL-PROCEDURE procedure files, ENTER files and files containing DBH load parameters, should be
  - protected by passwords
  - only deleted by specifying `DESTROY` in the `DELETE-FILE` command.

> **i** Provision must be made during system generation to ensure that UDS/SQL cannot be abnormally terminated as a result of password contraventions (see the manual "Introduction to System Administration" CL2-Option):
>
> PWACTIVE system parameter:
> The value must be at least one more than the number of passwords which can be attached concurrently by means of `PP PASSWORD` and `ADD PW.`
>
> PWENTERD system parameter:
> The value must be much larger than the number of passwords. If realms are to be frequently attached and detached in the course of a session, it is advisable to use the maximum value (= default value).
>
> Before files are attached or detached, UDS/SQL respectively always issues and revokes all known passwords.
>
> PWERRORS system parameter:
> The value should be much larger than the number of UDS/SQL files protected by the same password. Otherwise just a few errors in entering the password can result in UDS/SQL tasks being abnormally terminated by BS2000 (maximum value = default value).

9.11 Accounting in UDS/SQL openUTM operation

UDS/SQL supports the accounting interface provided by openUTM to the databases. The CPU utilization and the number of I/Os per openUTM job is recorded as long as the openUTM application requests accounting information.

The recording of accounting data itself consumes CPU time and should therefore be carried out according to the requirements of the application. Individual utilization values are highly dependent on the current status of the UDS /SQL session.

UDS/SQL also executes subfunctions asynchronously, which means that the values passed to openUTM do not represent the total utilization.

The CPU time consumed by the master task and the administrator task is not taken into account, neither are initialization or termination phases of the UDS/SQL session.

The number of I/Os is determined at the end of a openUTM transaction. I/0s that occur after the end of a openUTM transaction are not counted.

UDS-D takes into account only the local configuration; it does not report the total amount of CPU time utilized over a number of different systems.

9.12 UDS/SQL users and user groups

In UDS/SQL-openUTM applications, openUTM ensures that the user is identified correctly and passes the relevant information (openUTM user ID, openUTM host name, openUTM application name and KSET name) to UDS/SQL. UDS/SQL uses this information to construct the UDS/SQL user name and the UDS/SQL user group name as follows:

```
UDS/SQL user group name       =  openUTM host name +
                                 openUTM application name +
                                 KSET name
UDS/SQL user name             =  openUTM host name +
                                 openUTM application name +
                                 openUTM user ID
```

Underscore characters are added internally to some user names and user group names.

The UDS/SQL user group name is used for checking authorization; the UDS/SQL user name is of no importance in this version. In UDS/SQL openUTM applications, the assignment of UDS/SQL users to UDS/SQL user groups is superfluous. If it is necessary to change the assignment of users to given user groups, these changes only need be made by the openUTM administrator.

The load parameter `PP PRIVACY-CHECK` can be used by the UDS/SQL database administrator to modify or turn off the privacy check (see PRIVACY-CHECK in chapter "DBH load parameters").

To provide a clearer picture of the general organization of UDS/SQL and openUTM, the openUTM concept of access privileges is described briefly below.

9.12.1 Access control via openUTM

In openUTM applications, services requested by terminal users or client programs are implemented by means of program units. These program units contain the openUTM calls and any database accesses that may be required. Different access privileges may be differentiated within an application.

openUTM offers two access control methods for this purpose which provide the same differentiation options but differ in the way they view the application’s objects:

- the user-oriented lockcode/keycode concept and
- the role-oriented access list concept

Lockcode/keycode concept

The lockcode/keycode concept enables you to ensure that only specially authorized users of client programs may use particular services of the UTM application. You can also specify that it is only possible to log on under a user ID via specific LTERM partners (connection points) or that particular services can only be started via special LTERM partners.

The objects to be protected - for example LTERM partners and the transaction codes assigned to the services - can be assigned a lockcode.

Keycodes are defined for user IDs and LTERM partners. When a keycode matches the lockcode of a protected object, access to this object is permitted.

A user ID or an LTERM partner can generally access multiple services and consequently has multiple keycodes. The individual keycodes are therefore grouped in keysets.

The lockcode and keycode concept has the following effects:

- A terminal or client program can log on only if a keycode which matches the lockcode of the associated LTERM partner is allocated to the specified user ID.
- A terminal user or a client program can call a service only if **both** the keyset of the user ID concerned **and** that of the LTERM partner contain a keycode which matches the lockcode of the transaction code.

Access list concept

When the access list concept is used, users are grouped according to roles or functions in the company (gate keeper, clerical officer, human resources officer, head of department, administrator, controller, managing director, etc.); a user can naturally have multiple roles. Each role is mapped to a keycode.

- The administrator allocates one or more roles to each user of a UTM application (e.g. clerical officer, head of department, etc.).
- For the objects to be protected - services and TAC queues - an access list is then used to define which user groups (clerical officer, controller, etc.) have access.
- If, for example, you have selected *Human resources officer* as role 2 and *Managing director* as role 1, you can define that only these user groups should have access to the *Personnel* service by allocating the service an access list which contains codes 1 and 2.
- The users involved are in turn allocated a keyset which contains all roles (keycodes) which a user may have.

If you also want to define that data which is relevant to security may only be accessed via particular LTERM partners, you must also allocate appropriate keysets to the LTERM partners.

Access to a service then requires at least one role which is contained in the service’s access list to be defined for both the user and for the LTERM partner via whom the user is logged on.

Effect on UDS/SQL

Irrespective of the access control method used, all openUTM users who have the same keyset (KSET) in openUTM form a user group in terms of their database privileges. The privileges for each of these user groups which work with the database via UTM transaction codes must be defined in the UDS/SQL databases by means of the BPRIVACY utility routine.

For further information on access control, please refer to the openUTM manuals.

9.12.2 Defining UDS/SQL user groups and granting access privileges

Before the openUTM administrator generates a UDS/SQL-openUTM application, the UDS/SQL database administrator must define the UDS/SQL user group and its privileges. This is done using the `ADD-USER-GROUP` statement of the BPRIVACY utility routine.

Example

Three user groups are to be granted access to the SHIPPING database.
"User group 1" may perform openUTM administration duties.
"User group 2" is granted read and write access to the database.
"User group 3" is granted read access to the database.

The openUTM administrator generates the UDS/SQL-openUTM application "UTMAWVER" on the host "UTMHOST1" and makes the following specifications for the user groups:

User group 1

"UTMUSER1" with the password "PASSW001" may call the transaction codes that define the administration commands (prerequisite: the administration program KDCADM must be generated with KDCDEF). The KSET “KSETADM” is assigned to this user.
No database access takes place in this transaction code.

```
TAC KDCAPPL,ADMIN=YES,PROGRAM=KDCADM,LOCK=1
. . .
       Additional transaction codes for administration
. . .
TAC KDCUSER,ADMIN=YES,PROGRAM=KDCADM,LOCK=1
USER UTMUSER1,PASS=(C'PASSW001',DARK),KSET=KSETADM,PERMIT=ADMIN,STATUS=ON
KSET KSETADM,KEYS=1
```

User group 2

"UTMUSER2" and "UTMUSER3" with the passwords "PASSW002" and "PASSW003" may call the transaction codes "UPDATE", "ADD" and "SEARCH". The KSET “KSETUPD” is assigned to these users.

```
TAC UPDATE,ADMIN=NO,LOCK=2,PROGRAM=MODIFY
TAC ADD,ADMIN=NO,LOCK=3,PROGRAM=ADD-NEW
TAC SEARCH,ADMIN=NO,LOCK=4,PROGRAM=QUERY
USER UTMUSER2,PASS=(C'PASSW002',DARK),KSET=KSETUPD,STATUS=ON
USER UTMUSER3,PASS=(C'PASSW003',DARK),KSET=KSETUPD,STATUS=ON
KSET KSETUPD,KEYS={2,3,4}
```

User group 3

"UTMUSER4", "UTMUSER5" and "UTMUSER6" with the passwords "PASSW004", "PASSW005" and "PASSW006" may call the transaction code "SEARCH".
The KSET “KSETRTR” is assigned to these users.

```
TAC SEARCH,ADMIN=NO,LOCK=4,PROGRAM=QUERY
USER UTMUSER4,PASS=(C'PASSW004',DARK),KSET=KSETRTR,STATUS=ON
USER UTMUSER5,PASS=(C'PASSW005',DARK),KSET=KSETRTR,STATUS=ON
USER UTMUSER6,PASS=(C'PASSW006',DARK),KSET=KSETRTR,STATUS=ON
KSET KSETRTR,KEYS=4
```

When the above users issue database calls, the following UDS/SQL user group names result:

```
UTMHOST1UTMAWVERKSETUPD_
UTMHOST1UTMAWVERKSETRTR_
```

To enable the requests to be executed, the UDS/SQL database administrator must have defined these UDS/SQL user groups and their privileges **beforehand**, as shown in the following example:

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,     VERSION=02.9C00
/START-UDS-BPRIVACY
...
```

```
***** START BPRIVACY (UDS/SQL V2.9  1801 )     2019-01-29   09:26:52
% UDS0215 UDS STARTING UDS/SQL V2.9(LINKED-IN), DATE=2019-01-29 (ILL2038,09:26:52/4TE7)
% UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS (ILL1746,09:26:52/4TE7)
4TE7: UDS-PUBSET-JV: :IUDS:$XXXXXXXX.PUBSDECL.ALL
4TE7: PUBSETS: *
4TE7: DEFAULT PUBSET: IUDS
4TE7: ----------------------------------------
% UDS0722 UDS ORDER ADD RLOG 150628094054 IN EXECUTION (ILL1283,09:26:52/4TE7)
% UDS0356 UDS EXECUTION OF ORDERS FOR SHIPPING TERMINATED (ILL1309,09:26:52/4TE7)
```

```
//ADD-USER-GROUP USER-GROUP-NAME=*KSET-FORMAT(HOST=UTMHOST1,APPLICATION=UTMAWVER,
KSET=KSETADM),
  OBJECT=(*REALM(NAME=*ALL,RIGHT=ALL),*RECORD(NAME=*ALL,RIGHT=ALL),*SET(NAME=*ALL,
RIGHT=ALL))
//ADD-USER-GROUP USER-GROUP-NAME=*KSET-FORMAT(HOST=UTMHOST1,APPLICATION=UTMAWVER,
KSET=KSETUPD),
  OBJECT=(*REALM(NAME=*ALL,RIGHT=ALL),*RECORD(NAME=*ALL,RIGHT=ALL),*SET(NAME=*ALL,
RIGHT=ALL))
//ADD-USER-GROUP USER-GROUP-NAME=*KSET-FORMAT(HOST=UTMHOST1,APPLICATION=UTMAWVER,
KSET=KSETRTR),
 OBJECT=*REALM(NAME=*ALL,RIGHT=RETRIEVAL),*RECORD(NAME=*ALL,RIGHT=RETRIEVAL),*SET
(NAME=*ALL,RIGHT=RETRIEVAL))
//END
```

```
% UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE (ILL1758,09:27:25/4TE7)
4TE7: DATABASE NAME DMLS LOG READ PHYS READ LOG WRITE PHYS WRITE
4TE7: ------------------------------------------------------------------------
4TE7: SHIPPING                   13       112          59        42          20
% UDS0213 UDS NORMAL SYSTEM TERMINATION WITH **************13 DML-STATEMENTS 2019-01-29
(ILLY033,09:27:25/4TE7)
***** DIAGNOSTIC SUMMARY OF BPRIVACY

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END BPRIVACY (UDS/SQL V2.9  1801 )     2019-01-29   09:27:25
```

For further information, see the load parameter PP PRIVACY-CHECK in chapter "DBH load parameters" and the manual "Creation and Restructuring", BPRIVACY.

9.13 ACS (Alias Catalog Service)

When ACS is used in database tasks, aliases should never contain a catalog or user ID. Use of ACS is not permitted for the actual database files.

The ACS is effective only within a task. Consequently, the same alias may be defined for different files in different tasks.

Example

Task	Alias	File name
TASK1	ALIAS	FILE-1
TASK2	ALIAS	FILE-2
TASK3	ALIAS	FILE-3

Since UDS/SQL operates across multiple tasks, this could lead to inconsistencies in UDS/SQL databases.

If possible UDS/SQL checks on opening a database file whether the file name involved is an alias. If this is the case, an error message is output, and the corresponding action is aborted. In particular when aliases are used with a catalog and/or user ID (`COMPLETE-ALIAS-NAMES=*ALLOWED`) it is not possible to guarantee fully that this check is correct.

9.14 Unicode concept in UDS/SQL

Increasing internationalization means that the Unicode character set is vitally important, also in BS2000 and its applications. Detailed information about this is provided in the "Unicode in BS2000" manual.

Unicode combines all the text characters known worldwide in a single character set. Furthermore, Unicode is independent of the various vendors, systems and countries.

Unicode support in BS2000 is embedded in the existing concept for coded character sets (CCSs), see the "XHCS (BS2000)" manual.

A prerequisite for using Unicode is a suitable BS2000 system environment, the products COBOL2000 and CRTE currently being relevant for UDS/SQL, see the UDS/SQL Release Notice.

To support Unicode, UDS/SQL offers the NATIONAL data type. In COBOL this corresponds to the national items (PICTURE N(n), USAGE IS NATIONAL). COBOL and UDS/SQL support the Unicode variant UTF-16.

Details of this are provided in the "Design and Definition" manual and in the manuals on "COBOL2000 (BS2000)".

Use of Unicode (UTF-16) in existing UDS/SQL databases

The following options, among other things, are available to you for (partially) converting UDS/SQL databases to Unicode (UTF-16):

- Adding national items
- Adding a new record type
- Replacing existing items

Adding national items

You can proceed as follows to add national items:

- Add the data items in the schema and subschema DDL (see the "Design and Definition" manual).
  Define national items (Unicode, PICTURE N(n) USAGE IS NATIONAL) with the same number of characters as the existing alphanumeric items in the same record type. Please note that the length in bytes doubles here because a Unicode character occupies two bytes in UDS/SQL.

  You can also insert indicator fields to indicate the relevance of the new items.
- Restructure the database using BCHANGE/BALTER (see the "Creation and Restructuring" manual). BALTER initializes the new item with (national) blanks.
- In the records which are to be stored, fill the new item with national data. You can flag the old alphanumeric item as not relevant or empty or fill it with the previous alphanumeric data.

Adding a new record type

Instead of adding national items to existing record types you can also select the following procedure:

- Define a new record type
- Define the new national items which belong to an existing record type in the **new** record type
- Link the new record type as a member record type with the existing record type

The advantages of this procedure compared to adding new items are:

- Existing record types are not affected and consequently do not need to be changed

- The database can be restructured very quickly using BCHANGE/BALTER

The disadvantage is:

- Additional DML statements are required for the members. As a result the programming effort for the application increases and the performance deteriorates.

Replacing existing items

You can proceed as follows to replace existing items by national items:

- Change the data items affected in the schema and subschema DDL (see the "Design and Definition" manual). Replace an existing alphanumeric item by a national item (Unicode, PICTURE N(n) USAGE IS NATIONAL) with the same number of characters.
  Please note that the length in bytes doubles here because a Unicode character occupies two bytes in UDS/SQL.
- Unload the record types affected using BOUTLOAD (see the "Creation and Restructuring" manual)
- Restructure the database using BCHANGE/BALTER (see the "Creation and Restructuring" manual)
- If required, convert the BOUTLOAD output files from EBCDIC to Unicode using BS2000 products with an XHCS connection (e.g. PERCON or EDT)
- Load the unloaded record types using BINILOAD (see the "Creation and Restructuring" manual)

10 The SQL conversation

In an ESQL application, SQL-specific administration data is retained over transaction boundaries. This kind of data administration unit is called a conversation.

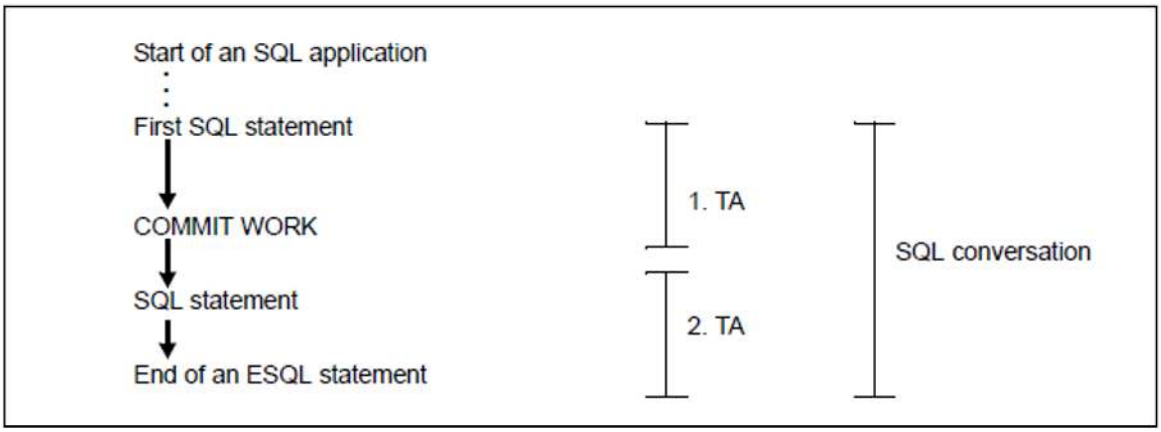


Figure 10: SQL conversation

The SQL conversation can assume various states:

- active conversation;
  a transaction of the ESQL application is open
- inactive conversation;
  no ESQL transaction is currently open.

During a conversation the DBH notes the declarations `DECLARE CURSOR` and `CREATE TEMPORARY VIEW`.

A conversation that has not been activated for a very long time may be terminated by the DBH or the database administrator. If this happens, the next ESQL statement will start a new conversation and the ESQL runtime system will again pass data on the conversation to the DBH.

See also the DBH load parameters PP SQL and PP SQL-LIMIT in chapter "DBH load parameters", and the DAL command DISPLAY SQL on "Displaying information on an SQL conversation (DISPLAY SQL)".

11 Output of database operating and status values with UDSMON

The UDS/SQL monitor UDSMON outputs values on database operation with UDS/SQL and UDS-D and on the status of transactions.
UDSMON works with the independent DBH only.
The following information can be output:

- current database performance
  - number of transactions per session and time interval
  - number of DML statements per session and time interval
  - number of SQL statements per session and time interval
  - average transaction times per session and time interval
  - number of input and output operations per session and time interval
- current status of active transactions
- use of system-internal optimization paths
  - inter-task communication (ITC)
  - probable position pointer (PPP)
- use of system-internal resources
  - number of open SQL conversations
  - number of active transactions
  - number of free server tasks
  - number and size of UDS/SQL common pools
- UDS-D operating values relating to
  - communication with remote partner configurations
  - current processing requests to remote configurations
  - current processing requests from remote configurations
  - connections to remote partner configurations

  UDS-D operating values are provided in
  - UDS-D fields of the COUNTER and STATUS masks
  - the UDS-D mask TRANSACTION
  - the UDS-D mask CONNECT

This information allows early detection of blockages resulting, for example, from longrunning jobs and deviations from normal system behavior.

Which values does UDSMON not provide?

UDSMON does not output values which can be provided by other system programs such as:

- SM2
  CPU loading, input/output loading for disks, channel loading, paging rate
- COSMOS
  CPU time, wait times, events

or by other UDS/SQL utility routines such as:

- BSTATUS

or by means of the system commands:

```
/SHOW-USER-STATUS;
/SHOW-FILE-ATTRIBUTES FILE-NAME=file-name [,SELECT=*ALL]
```

11.1 Description of functions

UDSMON reads UDS/SQL system tables in order to provide database operating values on system loading.

UDSMON outputs total counter readings, i.e. the total number of events which have occurred during the session. UDSMON also outputs rates, i.e. the number of events within a time interval.

Example

The DML rate is to be monitored during database operation at intervals of 10 seconds. If UDSMON is started at 12.00.00, the following specimen values may be provided:

Interval	**DML counter**	**DML rate**
12.00.00 start	80	0
12.00.00 - 12.00.10	90	10
12.00.10 - 12.00.20	120	30
12.00.20 - 12.00.30 . .	140 . .	20 . .

UDSMON reads the UDS/SQL management tables in order to provide transaction status values.

The status values which are output are valid for the relevant transaction at the end of the specified time interval.

Output media

The values obtained are output

- to a display terminal in formatted form
- to printer in formatted form
- to a file in unformatted form
- to openSM2/INSPECTOR: graphically edited

The time intervals can be specified and modified independently of each other for each medium. Each output medium can be activated or deactivated as required. This allows you to specify that the UDS/SQL monitor is to perform logging only at peak load times, for example.

External counter overflow

During conversion of the internal binary counters into printable decimal values, the output field may be too small to accommodate the counter.

In this case the counter is transferred left-aligned into the output field and, depending on the number of digits by which the counter has been truncated to the right, an appropriate letter is displayed in the column in front of the field.

D the counter has been truncated by one digit (decimal).
The value displayed must be multiplied by 10.

C the counter has been truncated by two digits (centesimal).
The value displayed must be multiplied by 100.

M the counter has been truncated by three digits (millesimal).
The value displayed must be multiplied by 1.000.

T the counter has been truncated by four digits.
The value displayed must be multiplied by 10.000.

H the counter has been truncated by five digits.
The value displayed must be multiplied by 100.000.

I the counter has been truncated by six digits.
The value displayed must be multiplied by 1.000.000.

? if the output field was more than six digits too small or an error occurred during conversion.

Internal counter overflow

When adding up counters from tables that occur more than once (e.g. per database or parallel transaction), overflows may occur in the internal sum fields. The monitor reports this with the following error message:

```
UDS0534 UDSMON INTERNAL OVERFLOW IN COUNTER counter-name
```

In addition, the following message appears in all monitor masks:

```
ATTENTION!! - INTERNAL OVERFLOW IN COUNTER counter-name --> MESSAGE UDS0534
```

counter-name

Identifies the mask field name of the counter involved.

The message remains in the masks until the next time you enter a `INFORM-PROGRAM` or `SEND-MSG` command.

11.2 System environment

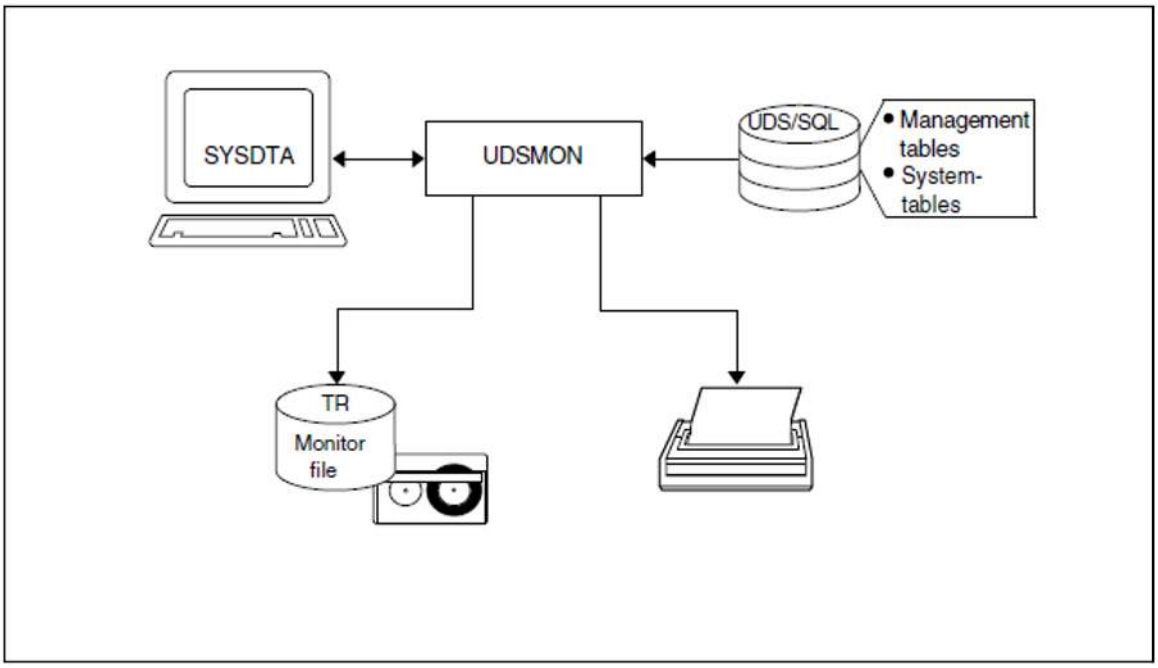


Figure 11: System environment of UDSMON

UDSMON is an autonomous program independent of UDS/SQL.

Starting the UDS/SQL monitor

UDSMON is started by:

```
/START-UDS-MONITOR
```

The UDS/SQL monitor must be loaded under the same user ID as the master task and server task(s). For further information on starting UDS utility routines, see the manual "Creation and Restructuring".

UDSMON accesses the UDS/SQL management and system tables. It executes parallel to database operation and outputs values obtained to the data display terminal, to printer or to a file.
Before the monitor starts to collect data, statements must be issued to specify the output devices, masks and time intervals.

If the database administrator wishes to specify a new or different output medium during UDSMON operation, an interrupt can be initiated by means of EM DUE followed by the appropriate SEND-MSG command.

Restriction:

If the UDS/SQL monitor attaches itself to the common pool, the version is checked. If the versions of UDS/SQL and UDSMON do not match, the monitor terminates with an appropriate message.

Terminating the UDS/SQL monitor

You can terminate the UDS/SQL monitor at any time with the following command:

```
/INFORM-PROGRAM MSG='END'[,JOB-ID=*OWN]
```

Otherwise it is terminated automatically at the end of the UDS/SQL session or once the time specified in the RUNTIME statement has expired.

Monitor file allocation

The appropriate CREATE-FILE command must be specified if the monitor file is to be set up explicitly.

```
/CREATE-FILE FILE-NAME=monitor-file
  [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary,SECONDARY-
ALLOCATION=secondary))
```

or

```
,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])]
/ADD-FILE-LINK LINK-NAME=TR,FILE-NAME=monitor-file,ACCESS-METHOD=*SAM
```

monitor-file

freely selectable name for the monitor file

TR link name of the monitor file

If no allocation is made, UDSMON automatically sets up the monitor file on public disk under the following name:

TR.*tsn*.*date*

TR specified file name (TR = TRACE)

tsn allocated task sequence number (four-digit)

date current date in the form: *yyyymmdd*

Any existing monitor file is updated.

11.3 Statements and commands for UDSMON

A distinction is made between UDSMON statements for starting and UDSMON commands during execution.

UDSMON statements for starting

Statement	Meaning
CONFNAME=*confname*	Allocates configuration
HELP	Lists all UDSMON statements
MASK={S | C | ALL | CS | DC | DT | D}	Assigns output mask(s) At the defined interval, UDSMON displays: S : the STATUS mask C : the COUNTER mask ALL : all masks in turn CS : the COUNTER mask and the STATUS mask alternately DC : the CONNECT mask DT : the TRANSACTION mask D : the TRANSACTION mask and the CONNECT mask alternately Default value : S
MEDIUM={T,*n* | L,*n*[,D] | F,*n*[,D] | S,*n*}	Assigns output device and time interval T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 Default value : T in interactive mode L in batch mode *n* : time interval, in seconds, between queries (*n*=5..999); for output to printer: between queries (*n*=1..999) Default value : 10 D : specifies that UDS-D operation is to be evaluated. With output to a file, an extra label and, at the set intervals, an extra data record are written. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is added.
DISPLAY={DEPRECATED | EXPENSIVE | MESSAGES | DBCOUNTERS}	Output to intermediate file (cf. corresponding command during operation, see table 27) The command is noted and executed at startup.
REBASE-COUNTER=NEW	Requests a logical zero point exclusively for the session counters of the counter mask

RUNTIME=*n*	Specifies the time after which UDSMON is to terminate automatically. This statement can be used particularly in batch mode, e.g. to obtain output to printer after a specified time. *n* : runtime in seconds (*n* = 60..86400) The time specified here should be greater than the time interval for the allocated output media.
START	Starts UDSMON Data collection commences.
END	Terminates UDSMON No data is collected.

Table 26: UDSMON statement for starting

UDSMON commands during execution

Execution of UDSMON can be interrupted to list or modify the specified devices, masks or time intervals.

Interruption is initiated by means of:
EM, DUE

followed by the command `INFORM-PROGRAM MSG = ... [, JOB-ID=*OWN'].`

Command	Meaning
INFORM**-PROG**RAM **MSG**='ADD MEDIUM= {T,*n* | L,*n*[,D] | F,*n*[,D] | S,*n*}'	Assigns output device and time interval T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 Default value : T in interactive mode L in batch mode *n* : time interval, in seconds, between queries (*n*=5..999); for output to printer: between queries (*n*=1..999) Default value : 10 D : specifies that UDS-D operation is to be evaluated. With output to a file, an extra label and, at the set intervals, an extra data record are written. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is added.
INFORM**-PROG**RAM **MSG**='DISPLAY'	Lists devices and time intervals currently set
INFORM**-PROG**RAM **MSG**='DISPLAY CP-SIZE'	Lists number and size of common pools

INFORM-**PROG**RAM **MSG**='DISPLAY DB-IO-COUNTER'	Lists all database I/Os that have occurred since starting the DBH and in the last time interval. The database I/Os are arranged in categories (2-Kbyte, 4-Kbyte, 8-Kbyte, and exclusive I/Os) and further subdivided into logical and physical read/write I/Os per category
INFORM-**PROG**RAM **MSG**='DISPLAY DEPRECATED'	List of applications with properties which contain a potential risk (limited address space utilization, obsolete loading technique, subschema validation missing) The list is written to the intermediate file (see section “DISPLAY output of the UDS monitor to intermediate files”) and displayed by the monitor by means of a SHOW-FILE command.
INFORM-**PROG**RAM **MSG**='DISPLAY EXPENSIVE'	List of applications with the greatest resource utilization with regard to transaction duration, number of DMLs per transaction, number of logical inputs/outputs per DML and per transaction, separated according to UTM and TIAM applications The list is written to the intermediate file (see section “DISPLAY output of the UDS monitor to intermediate files”) and displayed by the monitor by means of a SHOW-FILE command.
INFORM-**PROG**RAM **MSG**='DISPLAY MESSAGES'	Edited list of buffered DBH messages The list is written to the intermediate file (see section “DISPLAY output of the UDS monitor to intermediate files”) and displayed by the monitor by means of a SHOW-FILE command. In the case of consecutive commands in a monitor run with explicit allocation of the intermediate files, messages which have been added since the previous DISPLAY MESSAGES command are supplemented. Messages which are overwritten in the buffer and can therefore no longer be listed are tagged with a separate flag. The date change is displayed.
INFORM-**PROG**RAM **MSG**='DISPLAY DBCOUNTERS'	Displays the number of DMLs and I/Os per database since starting the DBH and while database is online. The list is written to the intermediate file (see section “DISPLAY output of the UDS monitor to intermediate files”) and displayed by the monitor by means of a SHOW-FILE command.
INFORM-**PROG**RAM **MSG**='DISPLAY REX'	Displays total number of transactions opened using READY EXCLUSIVE or READY PROTECTED
INFORM-**PROG**RAM **MSG**='DISPLAY SATURATION'	List of the greatest saturation of central system resources in the DBH session to date (e.g. number of parallel transactions, utilization of non-extensible memory pools)

INFORM**-PROG**RAM **MSG**='DISPLAY TASK'	Lists all the user tasks associated with UDS/SQL and the number of DMLs already processed in these tasks In the case of the processed DMLs, in contrast to comparable values of the counter mask, for diagnostic reasons requests of the COBOL runtime system which ensure correct transaction processing in the case of a task switch are also counted. For applications which were terminated asynchronously, e.g. as a result of timeouts in UTM applications, the following is also output: • whether this termination has been completed successfully and the internal UDS resources can be reused (RELEASED), • whether termination has not yet been fully completed (PENDING) or • whether the relevant internal UDS resource is locked permanently because of a serious error (BLOCKED).
INFORM**-PROG**RAM **MSG**='END'	Terminates the monitor
INFORM**-PROG**RAM **MSG**='FINISH MEDIUM={T | L[,D] | F[,D] | S}'	Terminates output on an output device T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 D : Only output for UDS-D operation is terminated. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is omitted. With output to a file, the UDS-D record is omitted.
INFORM**-PROG**RAM **MSG**='HELP'	Lists input options
INFORM**-PROG**RAM **MSG**='MASK={S | C | ALL | CS | DC | DT | D}'	Assigns output mask(s) At the defined interval, UDSMON displays: S : the STATUS mask C : the COUNTER mask ALL : all masks in turn CS : the COUNTER mask and the STATUS mask alternately DC : the CONNECT mask DT : the TRANSACTION mask D : the TRANSACTION mask and the CONNECT mask alternately Default value : S

INFORM**-PROG**RAM **MSG**='REBASE-COUNTER=NEW'	Rebasing of the counters of the COUNTER mask, i.e. requesting a logical zero point exclusively for the session counters of the COUNTER mask The next time the counters are collected, the current statuses of the COUNTER mask will be stored ("rebasing" or "snapshot" of the current counter statuses), and from then all counter statuses are specified with reference to this stored status (logical zero point). Rebasing can be used for the following in the case of extremely long DBH sessions: - to display comparable session counters, for example on a daily or weekly basis - to facilitate recognition of a change to counter statuses - to prevent problems in comparative interpretation when individual counters overrun Rebasing always refers to a single monitor instance. It is not possible to return to a previous rebasing. The values for a rebasing remain valid as long as the corresponding DBH session or the UDS monitor continues to run. The following events invalidate rebasing: - Restart of the DBH session When the DBH session restarts, the DBH's counters once more begin with 0. - Restart of the UDS monitor As the rebasing data is located locally in the monitor memory, it is lost when the monitor restarts.
INFORM**-PROG**RAM **MSG**='REBASE-COUNTER=OFF'	The counter statuses of the COUNTER mask which are used for rebasing are set to 0. As a result, the session counters in the COUNTER mask are once more displayed as absolute values with reference to the start of the DBH session.

Table 27: UDSMON commands during operation

11.4 DISPLAY output of the UDS monitor to intermediate files

In the case of the DISPLAY commands DISPLAY DEPRECATED, DISPLAY EXPENSIVE, DISPLAY MESSAGES and DISPLAY DBCOUNTERS the results are written to an intermediate SAM file which is displayed on the screen by means of a SHOW-FILE command which is issued internally by the UDS monitor.

When the results are displayed, you are in the output of the SHOW-FILE command and can use the convenient scroll and search functions of SHOW-FILE to evaluate the results. You terminate the DISPLAY command by entering END. You then quit the SHOW-FILE command, and UDSMON returns to its mask mode.

Link names of the intermediate files:

The intermediate file can be a permanent or temporary file and is addressed via the following link names:

$UDSDD in the case of DISPLAY DEPRECATED
$UDSDE in the case of DISPLAY EXPENSIVE
$UDSDM in the case of DISPLAY MESSAGES
$UDSDC in the case of DISPLAY DBCOUNTERS

If no file is assigned via these link names, UDSMON creates a temporary file with the following names:

#UDSMON.DEPRECATED
#UDSMON.EXPENSIVE
#UDSMON.DBH.MESSAGES
#UDSMON.DBCOUNTERS

where # is the temporary flag and can be changed to @ using the BS2000 system parameter TEMPFILE.

Only if the use of temporary files is not possible on the BS2000 system (BS2000 system parameter TEMPFILE='NO') does UDSMON create a permanent file with the following names:

UDSMON.DEPRECATED.*tsn*.*date*
UDSMON.EXPENSIVE.*tsn*.*date*
UDSMON.DBH.MESSAGES.*tsn*.*date*
UDSMON.DBCOUNTERS.*tsn*.*date*

Use of the intermediate files:

With each DISPLAY DEPRECATED, DISPLAY EXPENSIVE, DISPLAY MESSAGES and DISPLAY DBCOUNTERS command, UDSMON checks whether an intermediate file with the corresponding name is already assigned. If no such file exists, UDSMON creates a new intermediate file and addresses it using temporarily the relevant link name.

In the case of a DISPLAY EXPENSIVE, DISPLAY MESSAGES, or DISPLAY DBCOUNTERS command UDSMON opens the intermediate files in EXTEND mode if an intermediate file with the relevant link name is already assigned. In this case the files are updated. If multiple DISPLAY MESSAGES commands are issued in a monitor run, only the DBH messages which have been added since the previous command are supplemented. If no intermediate file with link names is assigned in the case of the DISPLAY EXPENSIVE, DISPLAY MESSAGES or DISPLAY DBCOUNTERS command or a new file name is recognized, the intermediate file is opened with OUTPUT mode.
In the case of the DISPLAY DEPRECATED command the intermediate file is opened in OUTPUT mode and consequently overwritten with each new DISPLAY command.

The intermediate files are only opened while a DISPLAY command is being processed. As a result you have many different options to intervene until another DISPLAY command is issued, e.g. you can rename the file, copy or also delete it if, in the case of updated files, the previous edition is no longer required.

DISPLAY commands, which write the result to an intermediate file, can also be entered as START commands. These commands are then noted and executed immediately after the START command. However, in this case no internal SHOW-FILE call is executed after the intermediate file has been written to.

Example of an output to an intermediate file

```
=== UDSMON: DISPLAY DBCOUNTERS ============================================
 DB-CONFNAME : FPADB                DATE:   2019-01-29     TIME:   09:26:07
 DATABASE NAME              DML   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 FPADB                        5         40         23         14          8
 DBSTAT                       0         26         20          2          2
=== UDSMON: END OF DISPLAY DBCOUNTERS =====================================
```

11.5 Command sequence for starting and operating UDSMON

```
01  [/CREATE-FILE FILE-NAME=monitor-file
      [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary -
       ,SECONDARY-ALLOCATION=secondary)) or
       ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])]]
    [/ADD-FILE-LINK LINK-NAME=TR,FILE-NAME=monitor-file,ACCESS-METHOD=*SAM]
02  /START-UDS-MONITOR
03  "udsmon statements for starting"
04 EM, DUE
05 "udsmon commands during execution"
06 EM, DUE
07 /INFORM-PROGRAM MSG='END'
```

Explanation

01 If the monitor file is to be set up explicitly, the `CREATE-FILE` command must be used for this purpose. In this case specifying :*catid*: is permitted in accordance with BS2000 conventions (see section “Using pubsets in UDS/SQL”).

02 The monitor is started.

04/06 An interrupt during monitor execution is initiated by means of EM, DUE.

07 The monitor is terminated.

After a program interrupt, due to [K2] for example, the `INFORM-PROGRAM` command is required in order to resume operation.

In batch mode, an invalid UDSMON statement causes the monitor start to be terminated. An error message is output to SYSOUT.

11.6 Example

The database operating values of the configuration EXAMPLE are to be monitored. The output mask is to contain status details. The first output device is to be a printer. The interrogation frequency for the printer is to be 60 seconds. The second output device is to be a data display terminal.

The interrogation cycle for the data display terminal is to be 30 seconds. The monitor is to terminate itself after 30 minutes.

```
/SET-JV-LINK LINK-NAME=VERS,JV-NAME=JV.MANUAL.VERSION
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=&(*VERS),SCOPE=*TASK
/START-UDS-MONITOR
...
%  UDS0501 UDSMON ENTER START PARAMETER OR 'HELP‘ (MO81446,13:25:51/66B2)
*CONFNAME=BEISPIEL
%  UDS0501 UDSMON ENTER START PARAMETER OR ,HELP‘ (MO81446,13:26:09/66B2)
*MASK=S
%  UDS0501 UDSMON ENTER START PARAMETER OR ,HELP‘ (MO81446,13:26:21/66B2)
*MEDIUM=L,60
%  UDS0501 UDSMON ENTER START PARAMETER OR ,HELP‘ (MO81446,13:26:35/66B2)
*MEDIUM=T,30
%  UDS0501 UDSMON ENTER START PARAMETER OR ,HELP‘ (MO81446,13:26:49/66B2)
*RUNTIME=1800
%  UDS0501 UDSMON ENTER START PARAMETER OR ,HELP‘ (MO81446,13:27:03/66B2)
*START
```

11.7 Description of UDS/SQL monitor output to data display terminal

The masks output on the data display terminal by the UDS/SQL monitor (UDS monitor masks STATUS and COUNTER and UDS-D monitor masks TRANSACTION and CONNECT) are described below.

Description of the UDS/SQL monitor mask: STATUS

```
UDS/SQL-MONITOR 2.9 * DB-CONFNAME: EXAMPLES                    DATE: 2019-01-29
STATUS              * INTERV. SEC: 030 TSKCON: 002 TAACT: 0002  TIME: 13:49:46
*********************
     TA     DML     SQLTA SQLDML RDMLO RDMLI LRDDB PRDDB LWRDB PWRDB PRDRL PWRRL
IV:  000000 0000000 00000 000000 00000 00000 00000 00000 00000 00000 00000 00000
PROGRAM  TSN  RUN-ID   MO ST DB-NAME  CALLS   DML  RDML  LRDDB PRDDB LWRDB PWRDB
IQS      3VKT 00000004 SR IO VERSAND  FTC4R   0006 0000  00012 00005 00000 00000
IQS      3VMT 00000005 SU US KUNDEN   READY   0001 0000  00000 00000 00000 00000
--------------------------------------------------------------------------------
LTG                                                          TAST
```

In the two header lines the UDS/SQL monitor outputs information on the database configuration, current date, output frequency and time:

DB-CONFNAME : *confname*

name of the DB configuration

DATE : *yyyy-mm-dd*

current date (year, month, day)

INTERV. SEC: *nnn*

time interval in seconds for mask output

TSKCON:

number of user tasks connected to UDS/SQL

TAACT:

number of active transactions at the end of an interval

TIME : *hh*:*mm*:*ss*

time (hour, minute, second)

The subsequent lines contain information on the internal counter values of the last output interval and on the transactions currently active.

Information on a maximum of 16 active transactions is displayed in this mask, sorted in increasing order of transaction identifier. In the event of more than 16 active transactions, the 16 oldest are output.

IV (interval):

TA number of transactions

DML number of CODASYL statements and SQL statements

SQLTA

number of SQL transactions

SQLDML

number of SQL statements

For UDS-D:

RDMLO

number of DML statements sent to remote partner configurations for processing

For UDS-D:

RDMLI

number of DML statements received from remote partner configurations for processing

LRDDB

number of logical read calls (`READ`) to the databases

PRDDB

number of physical read calls (`READ`) to the databases

LWRDB

number of logical write calls (`WRITE`) to the databases

PWRDB

number of physical write calls (`WRITE`) to the databases

PRDRL

number of physical read calls (`READ`) to the RLOG file

PWRRL

number of physical write calls (`WRITE`) to the RLOG file; if duplicate RLOG files are maintained, the calls are counted only once

For each active transaction, the following information is provided on the program(s) started:

PROGRAM

progname
In timesharing mode: name of the application program
In transaction mode: database key (openUTM)

For UDS-D:

If *progname* is followed by an '*', it designates a secondary subtransaction, i.e. the application program was started in another configuration.

TSN *tsn*
In timesharing mode: task serial number under which the program *progname* runs;
In transaction mode: task serial number of the last openUTM task used in the transaction.

RUN-ID

transaction-id

Internal transaction identifier assigned by UDS/SQL. Transactions are numbered consecutively within a session.

MO open mode of the transaction

```
SR       SHARED RETRIEVAL
SU       SHARED UPDATE
PR       PROTECTED RETRIEVAL
PU       PROTECTED UPDATE
ER       EXCLUSIVE RETRIEVAL
EU       EXCLUSIVE UPDATE
```

ST current transaction status

AR=AREA
The transaction is in the process of executing a READY statement in a realm queue; it is currently unable to access a requested realm.

CA=IN CANC
The CANCEL request is being processed or

CA=TO CANC
a CANCEL request for the transaction is pending

CC=CONNEC
Control of the transaction resides in the connection module

DH=DBH
DML is being processed by the DBH

IO=INPUT/OUTPUT
The transaction is waiting for the end of an asynchronous input or output operation

LO=LOCK
The transaction is waiting for a page lockout to be canceled

SM=SEMAPHOR
The transaction is waiting for a code lock to be canceled

SP=STOP
The transaction is in the STOP state or

SP=TOSTOP
a STOP request is pending

ST=SERVERTASK
The transaction is waiting for a free server task

US=USER
DML with the user

DB-NAME

dbname
Name of the database being processed by the current DML statement

CALLS

Short function names for DML and SQL statements (see DAL command DISPLAY SQL "Displaying information on an SQL conversation (DISPLAY SQL)" and section “Function codes of the DML statements”).

DML serial number of DML statement of the transaction:

For UDS-D:

- *progname* *:
  serial DML number of a secondary subtransaction (application program in another configuration)

- *progname* without *:

  serial DML number of a purely local transaction or, if RDML counter allocated,
  serial DML number of a primary subtransaction

RDML For UDS-D:
serial number of the remote DML statement sent to another configuration (remote DML)

LRDDB

number of logical read calls (`READ`) to the databases

PRDDB

number of physical read calls (`READ`) to the databases

LWRDB

number of logical write calls (`WRITE`) to the databases

PWRDB

number of physical write calls (`WRITE`) to the databases

Description of the UDS/SQL monitor mask: COUNTER

```
UDS/SQL-MONITOR 2.9 *   DB-CONFNAME : EXAMPLES             DATE    : 2019-01-28
COUNTER              *   MACHINE TYPE:  S190-40            TIME    : 13:13:16
*********************   MAIN MEMORY : 001232 MB           INTERVAL: 030 SEC
                    SESSION    INTERVAL                      SESSION    INTERVAL
SQL CONVERSATIONS   0000000000 00000000  LOGICAL  READ DB    0000000140 00000010
TRANSACTIONS        0000000005 00000000  PHYSICAL READ DB    0000000074 00000005
SQL TRANSACTIONS    0000000000 00000000  PHYSICAL READ RLOG  0000000000 00000000
UPDATE TRANSACT.    0000000000 00000000  LOGICAL  WRITE DB   0000000000 00000000
REMOTE UPD. TRANS.  0000000000 00000000  PHYSICAL WRITE DB   0000000000 00000000
DML CALLS           0000000049 00000005  PHYSICAL WRITE RLOG 0000000004 00000000
SQL CALLS           0000000000 00000000  PHYSICAL WRITE ALOG 0000000000 00000000
RDML OUT            0000000000 00000000
RDML IN             0000000000 00000000  DATA   DEADLOCKS    0000000000 00000000
                                         GLOBAL DEADLOCKS    0000000000 00000000
ITC ST --> US       0000000049 00000005  LOCK WAITS          0000000000 00000000
ITC US --> ST       0000000049 00000005
REQUEST ST --> ST   0000000022 00000005  PPP OK                   100.0%   ---.-%
ITC ST--> ST        0000000000 00000000  AVG TRANSACTION TIME   000.700s 000.300s
PP TRANSACTION      DEF  0004  ACT 0002  PP SQL               DEF  0004 ACT  0000
PP SEC. TRANSACT.   DEF  0001  ACT 0000  PP SERVERTASK        DEF  0001 FREE 0001
SEC. TRANSACT. OUT             ACT 0000  PP 2KB/4KB/8KB-BUFFERSIZE 0001/0001/0000
--------------------------------------------------------------------------------
LTG                                                        TAST
```

The three header lines contain information on the database configuration, current date and time, output frequency and hardware used.

DB-CONFNAME : *confname*

name of the database configuration

DATE : *yyyy-mm-dd*

current date (year, month, day)

MACHINE TYPE : *XXXXXXXX*

name of the system (up to eight characters), e.g. "S190-40"

TIME : *hh*:*mm*:*ss*

time (hour, minute, second)

MAIN MEMORY : *nnnn*.*n* MB

size in Mbytes of usable main memory

INTERVAL: *nnn*

time interval in seconds for mask output

The subsequent lines list information on internal counter values since the start of the session and during the last output interval:

SQL CONVERSATIONS

number of SQL conversations processed

TRANSACTIONS

number of CODASYL and SQL transactions

SQL TRANSACTIONS

number of SQL transactions

UPDATE TRANSACT.

number of update CODASYL and SQL transactions

REMOTE UPD. TRANS.

number of transactions with update in a remote partner configuration

DML CALLS

number of CODASYL statements and SQL statements

SQL CALLS

number of SQL statements

For UDS-D:

RDML OUT

number of DML statements sent to remote partner configurations for processing (remote DML)

For UDS-D:

RDML IN

number of DML statements received from remote partner configurations for processing

ITC ST --> US

number of performed inter-task communications (ITC) from the server task (ST) to the user (US)

ITC US --> ST

number of performed inter-task communications (ITC) from the user (US) to the server task (ST)

REQUESTS ST --> ST

sum of all requests from server task (ST) to server task

ITC ST --> ST

number of performed inter-task communications (ITC) from server task (ST) to server task

LOGICAL READ DB

number of logical read calls to the databases (READ)

PHYSICAL READ DB

number of physical read calls to the databases (READ)

PHYSICAL READ RLOG

number of physical read calls to the RLOG file

LOGICAL WRITE DB

number of logical write calls to the databases (WRITE)

PHYSICAL WRITE DB

number of physical write calls to the databases (WRITE)

PHYSICAL WRITE RLOG

number of physical write calls to the RLOG file; with duplicate RLOG backup, calls are counted only once

PHYSICAL WRITE ALOG

number of physical write calls to the ALOG file

DATA DEADLOCKS

number of data deadlocks

GLOBAL DEADLOCKS

number of task deadlocks during openUTM operation and interconfiguration deadlocks (if UDS-D is used)

LOCK-WAITS

number of lock waits

PPP OK

number of successful attempted accesses via probable position pointer (PPP) in percent

AVG TRANSACTION TIME

average transaction times in seconds, e.g. 0.099 sec

The following lines provide information on the DBH load parameters, as defined (DEF) on starting the DBH, and the extent to which the requested resources are being used (FREE/ACT):

PP TRANSACTION DEF

maximum possible number of simultaneously active transactions = value in PP TRANSACTION

For UDS-D:
primary and secondary subtransactions and local transactions

PP TRANSACTION ACT

number of active transactions

For UDS-D:
primary and secondary subtransactions and local transactions currently being processed by this DBH

PP SEC. TRANSACT. DEF

For UDS-D:
maximum possible number of secondary subtransactions that can be processed by this DBH (value *m* in PP TRANSACTION)

PP SEC. TRANSACT. ACT

For UDS-D:
number of active secondary subtransactions (STTs) currently being processed by this DBH

SEC. TRANSACT. OUT ACT

For UDS-D:
number of active secondary subtransactions (STTs) if the application program is in its own configuration

PP SQL DEF

number of SQL conversations defined with PP SQL

PP SQL ACT

number of active SQL conversations

PP SERVERTASK DEF

number of server tasks defined with PP SERVERTASK

PP SERVERTASK FREE

number of free server tasks

PP 2KB/4KB/8KB-BUFFER-SIZE

size of system buffer pools in Mbytes; as specified for PP 2KB-BUFFER-SIZE, PP 4KB-BUFFER-SIZE and PP 8KB-BUFFER-SIZE, respectively.

Description of the UDS-D monitor mask: TRANSACTION

```
UDS/SQL-MONITOR 2.9 * DB-CONFNAME: EXAMPLES                    DATE: 2019-01-29
UDS-D   TRANSACTION * INTERVAL   : 030 SEC     PARTNER: 0002    TIME: 14:03:27
*********************
IV:  TA     DML     SQLTA SQLDML       LRDDB PRDDB  LWRDB PWRDB  PRDRL PWRRL
     000045 0000612 00000 000000       01024 00000  00260 00000  00000 00040
SUM:           STT-OUT          STT-IN           RDML-OUT         RDML-IN
           SESSION  INTERV  SESSION  INTERV   SESSION  INTERV  SESSION  INTERV
           00000390 000022  00000620 000022   00005455 000306  00008646 000302
CONFNAM
BEISP02    00000390 000022  00000620 000022   00005455 000306  00008646 000302
BEISP01    00000000 000000  00000000 000000   00000000 000000  00000000 000000
-------------------------------------------------------------------------------
LTG                                                          TAST
```

In the two header lines the UDS/SQL monitor outputs information on the database configuration, current date, output frequency, number of partner configurations, and time:

DB-CONFNAME : *confname*

name of the home DB configuration

DATE : *yyyy-mm-dd*

current date, specified as year, month, day

INTERVAL: *nnn*

time interval in seconds for mask output

PARTNER

number of partner configurations known to this configuration

TIME: *hh-mm-ss*

time, specified as hour, minute, second

The following lines contain information on the internal counter values of the last output interval.

Information on a maximum of 13 partners is displayed in this mask.

First there is a list of the partners in which or by which DML statements were processed in the most recent output interval, followed by a list of all other known partners.

IV (interval):

TA	number of transactions
DML	number of CODASYL statements and SQL statements
SQLTA	number of SQL transactions
SQLDML	number of SQL statements
LRDDB	number of logical read calls (READ) to the databases

PRDDB number of physical read calls (`READ`) to the databases

LWRDB number of logical write calls (`WRITE`) to the databases

PWRDB number of physical write calls (`WRITE`) to the databases

PRDRL number of physical read calls (`READ`) to the RLOG file

PWRRL number of physical write calls (`WRITE`) to the RLOG file; with duplicate RLOG backup, calls are counted only once

The following lines contain information concerning the sum of all partner configurations.
The counter values since the start of the session and during the last interval are output.

STT-OUT

total number of secondary subtransactions sent to remote partner configurations for processing

STT-IN

total number of secondary subtransactions sent from remote partner configurations for processing

RDML-OUT

number of DML statements sent to remote partner configurations for processing (remote DML)

RDML-IN

number of DML statements received from remote partner configurations for processing

The following lines contain information on the attached partner configurations:

CONFNAM

name of the partner configuration

STT-OUT

total number of secondary subtransactions sent to this partner configuration for processing

STT-IN

total number of secondary subtransactions received from this partner configuration for processing

RDML-OUT

number of DML statements sent to this partner configuration (remote DML)

RDML-IN

number of DML statements received from this partner configuration for processing

Description of the UDS-D monitor mask: CONNECT

```
UDS/SQL-MONITOR 2.9 * DB-CONFNAME: EXAMPLES                    DATE: 2019-01-29
UDS-D   CONNECT     * INTERVAL   : 030 SEC     PARTNER: 0002    TIME: 12:13:24
*********************
IV:  TA     DML     SQLTA SQLDML RDMLO RDMLI LRDDB PRDDB LWRDB PWRDB PRDRL PWRRL
     000047 0000657 00000 000000 00333 00322 01103 00000 00286 00000 00000 00044

SUM:          CONNECT-OUT            CONNECT-IN        DISCON    GLOB-DEADLOCKS
         REQUEST REJECT ACTIVE  REQUEST REJECT ACTIVE  SESS      SESS
         0000005 000000 000002  0000007 000000 000000  000000    00000000000000
CONFNAM
BEISP02  0000005 000000 000002  0000007 000000 000000  000000
BEISP01  0000000 000000 000000  0000000 000000 000000  000000
--------------------------------------------------------------------------------
LTG                                                          TAST
```

In the two header lines the UDS/SQL monitor outputs information on the database configuration, current date, output frequency, number of partner configurations, and time:

DB-CONFNAME : *confname*

name of the home DB configuration

DATE : *yyyy-mm-dd*

current date, specified as year, month, day

INTERVAL: *nnn*

time interval in seconds for mask output

PARTNER

number of partner configurations known to this configuration

TIME: *hh-mm-ss*

time, specified as hour, minute, second

The next lines contain information on the internal counter values of the last output interval.

Information on a maximum of 13 partners is displayed in this mask.

First there is a list of the partners to which there are currently active connections, followed by a list of all other known partners.

IV (interval):

TA number of transactions

DML number of CODASYL statements and SQL statements

SQLTA

number of SQL transactions

SQLDML

number of SQL statements

RDMLO

number of DML statements sent to remote configurations for processing

RDMLI

number of DML statements received from remote application programs for processing

LRDDB

number of logical read calls (`READ`) to the databases

PRDDB

number of physical read calls (`READ`) to the databases

LWRDB

number of logical write calls (`WRITE`) to the databases

PWRDB

number of physical write calls (`WRITE`) to the databases

PRDRL

number of physical read calls (`READ`) to the RLOG file

PWRRL

number of physical write calls (`WRITE`) to the RLOG file; with duplicate RLOG backup, calls are counted only once

The next lines contain information relating to the sum of all partner configurations. The counter values since the start of the session are output.

CONNECT-OUT

REQUEST
total number of connection requests issued to remote partner configurations in the course of the session

CONNECT-OUT

REJECT
total number of connection requests rejected by remote partner configurations

CONNECT-OUT

ACTIVE
total number of currently active connections to remote partner configurations requested by the home configuration

CONNECT-IN

REQUEST
total number of connection requests issued by remote partner configurations to the home configuration in the course of the session

CONNECT-IN

REJECT
total number of connection requests from remote partner configurations rejected by the home configuration

CONNECT-IN

ACTIVE
total number of currently active connections to the home configuration requested by remote partner configurations

DISCON SESS

total number of connections to remote partner configurations cleared down by application programs of the local configuration in the course of the session due to errors. Reasons for disconnection:

- CHECKTIME monitoring by the user
- errors reported by DCAM

GLOB-DEADLOCKS SESS

total number of task deadlocks detected by the lock manager of the home configuration, i.e. task deadlocks that occur during openUTM operation and interconfiguration deadlocks when UDS-D is used. There is assumed to be an interconfiguration deadlock when interconfiguration contention situations between distributed transactions are not resolved within a specified time. This time can be set with PP DEADTIME.

The following lines contain information on the attached partner configurations. The counter values since starting the DBH are output.

CONFNAM

name of the partner configuration

CONNECT-OUT

REQUEST
total number of connection requests issued to this partner configuration in the course of the session

CONNECT-OUT

REJECT
total number of connection requests rejected by this partner configuration in the course of the session

CONNECT-OUT

ACTIVE
total number of active connections to this partner.

A connection can be made to this partner configuration from any application program and from the UDS-D task of the home configuration. The home/remote UDS-D task connection is counted only in CONNECT-IN ACTIVE.

CONNECT-IN

REQUEST
total number of connection requests issued by this partner configuration in the course of the session

CONNECT-IN

REJECT
total number of connection requests from this partner configuration rejected by the home configuration

CONNECT-IN

ACTIVE
total number of currently active connections from this partner configuration to the home configuration.

A connection can be made from this partner configuration from any application program and from the remote UDS-D task.

DISCON SESS

total number of connections to this partner configuration cleared down in the course of the session due to errors

11.8 Description of UDS/SQL monitor output to printer

When a list is output to a printer, the counter values in the COUNTER and STATUS masks are combined.

In addition, information on UDS-D can be output from the TRANSACTION and CONNECT monitor masks. This output is controlled by means of the UDSMON command `MEDIUM=L,n,D` (D=distributed).

Generally, the content of the list output corresponds to the output at the data display terminal (see chapter "Description of UDS/SQL monitor output to data display terminal"). There are, however, the following differences:

- In the case of the STATUS, TRANSACTION and CONNECT monitor masks there is no interval counter (IV) information line. Information on all active transactions is output.
- In the case of openUTM transactions, the "*" character is output after the TSN in the TSN column of the STATUS mask.
- If a transaction in OSI TP is distributed globally with "Functional Unit Commit", up to 64 bytes of the Atomic Action Identifier (AAID) are output in hexadecimal format for this transaction in an additional line of the list output. In the case of transactions which are distributed via UDS-D, output takes place only in the configuration of the primary subtransaction, in other words in the configuration in which the UTM application concerned is connected. The AAID is generated by an OSI-TP application which controls the transaction as a partner of a UTM application.
  This controlling OSI-TP application can be for example UTM (on BS2000 or other platforms), the LU6.2 gateway or the generic BeanConnect-Proxy UTM application on Linux
  This AAID is generated in the OSI-TP gateway at the start of the transaction (e.g. in theopenUTM-LU62 gateway).
  Generation of the AAID is logged in the gateway’s corresponding instance trace. This enables the transaction to be assigned to the controlling transaction in the foreign system.

11.9 Description of UDS/SQL monitor output to a file

In response to the UDSMON command `MEDIUM=F,`*n*`,[D]` the UDS/SQL monitor writes data records to the output file after each specified time interval.
A label is written to the file when UDSMON output starts or if the time interval is changed.

The output file is a SAM file with variable-length records. The current counter readings are output in binary form.

For UDS-D:
If D (=distributed) is entered during operation in UDS-D mode, UDS-D operation is evaluated as well.

In such cases the structure of the output file is as follows:
UDS/SQL label
UDS-D label
UDS/SQL data record
UDS-D data record
. . .

The UDS-D label is output when UDSMON output starts, if the time interval is changed, if the DBH load parameter PP PTCSYNCH is changed, or if the distribution table is activated or deactivated.

> ¡ The record format may change in a subsequent version of UDS/SQL. As a rule this will entail increasing the record length.
>
> The UDS/SQL version number appears in label field 4 (see the table “UDS/SQL label format”).

Record format for file output

UDS/SQL label format

Field number	Field contents		Displacement [dec]	Length [bytes]
1	Record length		0	2
2	Filler		2	2
3	Record identifier X'0010'		4	2
4	UDS/SQL version number X'0290' =V2.9		6	2
5	Time in STCK format		8	8
6	Configuration name		16	20
7	Time interval in seconds		36	4
8	PP SERVERTASK		40	2
9	PP TRANSACTION		42	2
10	PP 2KB-BUFFER-SIZE		44	2
11	PP MAXDB		46	2
12	PP SUBSCHEMA		48	2
13	PP LOG	X'00' no RLOG	50	1
		X'01' single RLOG		
		X'02' duplicate RLOG		
14	PP CPU	X'00' CPU-MONO	51	1
		X'01' CPU-MULTI		
15	PP DEACT	X'00' DEACTIVE=YES	52	1
		X'01' DEACTIVE=NO		
16	PP SCHEDULING	X'00' SYMMETRIC	53	1
		X'01' ASYMMETRIC		
17	PP IO	X'00' ASYNCHRON	54	1
		X'01' SYNCHRON		
18	Filler		55	1

19	Processor type, e.g. C'S150-60'	56	8
20	Internal identifier for the operating system and the operating system version	64	4
21	DCAM processor name (e.g. D016ZE09)	68	8
22	Main memory size in Kbyte, printable	76	8
23	Number of processors, printable	84	2
24	up to 8 serial numbers (binary) 3-byte identification number for each CPU (bytes 0-2: number of first CPU, bytes 3-5: number of second CPU, ...)	86	24
25	PP SQL	110	2
26	Filler (for internal use)	112	4
27	PP 4KB-BUFFER-SIZE	116	2
28	PP 8KB-BUFFER-SIZE	118	2
29	Date (printable) Format: *yyyy-mm-dd*	120	10
30	Time (printable) Format: *hh*:*mm*:*ss*	130	8
31	Values for the local time zone Format: *shh* $_1$:*mm* $_1$-*hh* $_2$:*mm* $_2$-*t* *s*: sign (“+“ or “-” for the time difference between local time zone and UTC time (Greenwich time) *hh* $_1$:*mm* $_1$: time difference between local time zoneand UTC time in hours (*hh* $_1$) and minutes (*mm* $_1$) *hh* $_2$:*mm* $_2$: time difference between daylight savingtime and normal time (i. e. summer and winter time) in hours (*hh* $_2$) and minutes (*mm* $_2$) *t*: current time setting for the local time zone(”W” for winter time, “S” for summer time)	138	14

Table 28: UDS/SQL label format

UDS-D label format

Field number	Field contents		Displacement [dec]	Length [byte]
1	Record length		0	2
2	Filler		2	2
3	Record identifier	X'0020'	4	2
4	UDS-D version number	X'0290' =V2.9	6	2
5	Time in STCK format		8	8
6	Time interval in seconds		16	4
7	PP TRANSACTION LOCAL or GLOBAL		20	2
8	PP TRANSACTION simultaneous STTs		22	2
9	PP CHCKTIME in seconds		24	2
10	PP DEADTIME in seconds		26	2
11	PP DISDB		28	2
12	PP PTCSYNCH-WARM	X'00' WAIT X'01' ABORT X'02' COMMIT	30	1
13	PP PTCSYNCH-SESSION	X'00' WAIT X'01' ABORT X'02' COMMIT	31	1
14	PP DISTABLE	X'00' no distribution table X'01' distribution table	32	1
15	Filler		33	3
16	Date (printable). Format: *yyyy-mm-dd*		36	10
17	Time (printable). Format: *hh*:*mm*:*ss*		46	8

18	Values for the local time zone. Format: *shh* $_1$:*mm* $_1$-*hh* $_2$:*mm* $_2$-*t* *s*: sign (“+“ or “-“ for the time difference between local time zone and UTC time (Greenwich time) *hh* $_1$:*mm* $_1$: time difference between local time zone and UTCtime in hours (*hh* $_1$) and minutes (*mm* $_1$) *hh* $_2$:*mm* $_2$: time difference between daylight saving time andnormal time (i. e. summer and winter time) in hours (*hh* $_2$)and minutes (*mm* $_2$) *t*: current time setting for the local time zone (”W” for winter time, “S” for summer time)	54	14

Table 29: UDS-D label format

UDS/SQL data record format

Field number	Field contents	Displacement [dec]	Length [bytes]
1	Record length	0	2
2	Filler	2	2
3	Record identifier X' 0011'	4	2
4	Filler	6	2
5	Time in STCK format	8	8
6	Free server tasks	16	4
7	Active transactions	20	4
8	Active SQL conversations	24	4
9	Number of DML statements	28	4
10	Number of transactions	32	4
11	Number of update transactions	36	4
12	Number of data deadlocks	40	4
13	Number of global task deadlocks	44	4
14	Filler (assigned the value 0)	48	4
15	Number of PPPs	52	4
16	Number of PPP OK	56	4
17	Number of LOCK-CALLs	60	4
18	Number of LOCK-Waits	64	4
19	Number of US --> ST requests	68	4
20	Number of ITC US --> ST	72	4
21	Number of ST --> ST requests	76	4
22	Number of ITC ST --> ST	80	4
23	Number of ITC ST --> US	84	4
24	Number of LOG. READ DBs	88	4
25	Number of LOG. WRITE DBs	92	4

26	Number of PHYS. READ DBs	96	4
27	Number of PHYS. WRITE DBs	100	4
28	Number of LOG. READ RLOGs	104	4
29	Number of LOG. WRITE RLOGs	108	4
30	Number of PHYS. READ RLOGs	112	4
31	Number of PHYS. WRITE RLOGs	116	4
32	Number of PHYS. READ ALOGs	120	4
33	Number of PHYS. WRITE ALOGs	124	4
34	Number of LOG. READ ALOGs	128	4
35	Number of LOG. WRITE ALOGs	132	4
36	Number of SEQUENTIAL READ DBs	136	4
37	Number of ACCEPT CRU without ITC	140	4
38	Number of SQLDMLs	144	4
39	Number of BIBs generated from SQLDML	148	4
40	Number of SQL transactions	152	4
41	Number of remote UPDATE transactions	156	4
42	Number of processed SQL conversations	160	4
43	Filler	164	4
44	Sum of TA times in seconds (binary)	168	4
45	Number of monitored transactions	172	4
46	Number of LOG. READ 2KB-DBs	176	4
47	Number of LOG. WRITE 2KB-DBs	180	4
48	Number of PHYS. READ 2KB-DBs	184	4
49	Number of PHYS. WRITE 2KB-DBs	188	4
50	Number of LOG. READ 4KB-DBs	192	4
51	Number of LOG. WRITE 4KB-DBs	196	4
52	Number of PHYS. READ 4KB-DBs	200	4
53	Number of PHYS. WRITE 4KB-DBs	204	4
54	Number of LOG. READ 8KB-DBs	208	4

55	Number of LOG. WRITE 8KB-DBs	212	4
56	Number of PHYS. READ 8KB-DBs	216	4
57	Number of PHYS. WRITE 8KB-DBs	220	4
58	Number of LOG. READ EXCL-DBs	224	4
59	Number of LOG. WRITE EXCL-DBs	228	4
60	Number of PHYS. READ EXCL-DBs	232	4
61	Number of PHYS. WRITE EXCL-DBs	236	4
62	Filler	240	10
63	Date (printable) Format: *yyyy-mm-dd*	250	10
64	Time (printable) Format: *hh*:*mm*:*ss*	260	8

Table 30: UDS/SQL data record format

UDS-D data record format

Field number	Field contents	Displacement [dec]	Length [bytes]
1	Record length	0	2
2	Filler	2	2
3	Record identifier X'0021'	4	2
4	Filler	6	2
5	Time in STCK format	8	8
6	STT ACTIVE OUT value	16	4
7	STT ACTIVE IN value	20	4
8	CONNECT ACTIVE OUT value	24	4
9	CONNECT ACTIVE IN value	28	4
10	Filler	32	4
11	Filler	36	4
12	Filler	40	4
13	Filler	44	4
14	Number of partners	48	4
15	RDML OUT value	52	4
16	RDML IN value	56	4
17	STT OUT value	60	4
18	STT IN value	64	4
19	CONNECT REQUEST OUT value	68	4
20	CONNECT REJECT OUT value	72	4
21	CONNECT REQUEST IN value	76	4
22	CONNECT REJECT IN value	80	4
23	DISCONNECT value	84	4
24	GLOB-DEADLOCKS SESS value	88	4
25	Filler	92	4

26	Date (printable) Format: *yyyy-mm-dd*	98	10
27	Time (printable) Format: *hh*:*mm*:*ss*	108	8

Table 31: UDS-D data record format

11.10 Transfer of monitor counters to openSM2

You can transfer the counter statuses of the UDS monitor's counter mask to openSM2 and present them there in graphical form using, for example, INSPECTOR. Some of these counter statuses can also be output on the terminal using the UDS SQL report of openSM2 or be evaluated in the SM2R1 reports.

A detailed description of openSM2 is provided in the "openSM2 (BS2000)" manual.

Prerequisite in UDS/SQL

When the UDS monitor is started, you initiate the transfer of the counter statuses in the counter mask to openSM2 using the following statement:

```
MEDIUM=S,n
```

During operation (see "Statements and commands for UDSMON") you start the transfer of the counter statuses to openSM2 with the command

```
INFORM-PROGRAM MSG='ADD MEDIUM=S,n'
```

You terminate the transfer of the counter statuses with the command

```
INFORM-PROGRAM MSG='FINISH MEDIUM=S'
```

n defines the interval in seconds (5 <= n <= 999) at which the UDS monitor supplies the data to SM2. It should be considerably lower than the measuring interval set in SM2 so that data can be transferred several times within an SM2 measuring interval.

The counter statuses can be transferred to openSM2 by means of a monitor instance which also initiates other outputs (outputs on the terminal, to a list or, in a special format, to a file). However, to guarantee uninterrupted supply it is recommended that you use a separate monitor run in the background to transfer the counter statuses to openSM2. When monitor instances run in the foreground, there is a risk that interruptions to change output media, masks or intervals will have undesired repercussions on the presentation of the counter statuses in openSM2.

Prerequisite in BS2000

A prerequisite for transferring data to openSM2 is that the receipt of the data is activated in the collector in SM2 with the following statement:

```
START-MEASUREMENT-PROGRAM TYPE=*UDS-SQL
```

If this interface is not activated or was deactivated with the `STOP-MEASUREMENT-PROGRAM TYPE=*UDS-SQL` statement, UDS/SQL issues a corresponding error message (UDS0536) once only.
However, an attempt is still made to supply openSM2 with the data so that after the UDS-SQL collecting program has been started the data can be transferred to openSM2 without any further measures being required in the UDS monitor.

To use the transferred data in INSPECTOR it is necessary for the INSPECTOR's agent to be active in BS2000 (`START-INSPECTOR`).

> The measured values are supplied asynchronously to openSM2 by UDS/SQL and apply for one or more intervals defined by UDS/SQL which need not precisely match the SM2 interval. Here differences can exist both in the length of the intervals and in the temporal displacements between the UDS/SQL and the SM2 intervals.
> The duration of one or more UDS/SQL intervals is used to standardize the measurement intervals to within a second. The values are therefore exact, but they only match the SM2 interval to a certain degree.

Displaying the counter with INSPECTOR

The INSPECTOR component of openSM2 enables you to present the data of the counter mask in graphical format on a workstation and to display it together with other usage values of the BS2000 system. In the UDS-SQL report group you select which counter statuses are to be displayed. At all times you can trace the development of the counter statuses over time, and thus assess the system behavior and detect problems at an early juncture.

In particular you can monitor the data on the basis of user-defined rules and thus recognize unusual statuses. In a rule conditions and actions can be specified. If the conditions apply, the associated actions are performed, e.g. SNMP traps can be triggered which lead to messages being sent to a connected SNMP station via the SM2 performance subagent, or emails relating to events can be sent.

A description of how to use INSPECTOR and a list of UDS/SQL counters which can be displayed there is provided in the "openSM2 (BS2000)" manual. A description of the SM2 performance subagent is provided in the "SNMP Management (BS2000)" manual.

Outputting the UDS SQL report of openSM2

You can also output some counter statuses of the counter mask on the BS2000 terminal using the UDS SQL report of openSM2. A detailed description of the UDS SQL report of openSM2 is provided in the "openSM2 (BS2000)" manual.

12 General functions of the utility routines

This chapter describes the general functions of the utility routines:

- Starting utility routines
- Checking the UDS/SQL pubset declaration
- Automatic realm extension
- Outputting database information in a neutral format (output in CSV format)
- Setting and evaluating job switches

12.1 Starting utility routines

The UDS-SQL utility routines are loaded and started using the following commands:

START-UDS-*program*
VERSION = **<u>*STD</u> /** <product-version> ,**MONJV** = **<u>*NONE</u>** / <filename 1..54 without-gen-vers> ,**CPU-LIMIT** = **<u>*JOB-REST</u>** / <integer 1..32767 *seconds*>

program
Name of the utility routine:
ADM, BALTER, BCALLSI, BCHANGE, BCHECK, BCREATE, BFORMAT, BGSIA, BGSSIA, BINILOAD, BMEND, BMODTT, BOUTLOAD, BPGSIZE, BPRECORD, BPRIVACY, BPSIA, BPSQLSIA, BRENAME, BREORG, BSTATUS, DDL, DMLTEST, MONITOR, ONLINE-PRIVACY or SSL.

VERSION =
Product version of the utility routine which is to be started.

VERSION = <u>*STD</u>
No explicit specification of the product version. The product version is selected as follows:

1. The version predefined with the /SELECT-PRODUCT-VERSION command.
2. The highest version installed with IMON.

VERSION = <product-version>
Explicit specification of the product version in the form mm.n[a[kk]].
You are recommended always to specify the version in full, e.g. 02.9C00, in order to facilitate migration in the event of correction packages.

MONJV =
Specifies a monitor job variable to monitor the DBH run.

MONJV = <u>*NONE</u>
No monitor job variable is used.

MONJV = <filename 1..54 without-gen-vers>
Name of the job variable to be used.

During the program run the system sets the job variable to the following values:

$R	Program running
$T	Program successfully terminated
$A	Program terminated with error

CPU-LIMIT =
Maximum CPU time in seconds which the program may take to execute.

CPU-LIMIT = <u>*JOB-REST</u>
The remaining CPU time for the BS2000 job is to be used for the task.

CPU-LIMIT = <integer 1..32767 *seconds*>
Only the time specified should be used.

12.2 Assigning the database name

The utility routines each process one database. The name of the database can generally be transferred by means of the link name DATABASE. When assignment takes place using ADD-FILE-LINK or SET-FILE-LINK, the file name of the DBDIR of the database concerned can also be specified as FILE-NAME, e.g. in order to identify the shadow database unambiguously.

Especially in the case of utility routines which use the linked-in DBH, the database name is also the configuration name. This name is also transferred by means of the link name DATABASE (see link assignment DATABASE for the configuration name of the DBH starting on "Starting DBH").

When an online utility routine (UDS online utility, ONLINE-PRIVACY) is used, the link name DATABASE is used only to assign the name of the DBH configuration. The database which must be processed here is determined using the OPEN-DATABASE statement or by means of an unambiguous subschema name.

12.3 Checking the UDS/SQL pubset declaration

When a utility routine starts, any UDS/SQL pubset declaration which exists is checked. The settings of an error-free UDS/SQL pubset declaration are recorded for further use, but they are not logged.

> ¡ A faulty job variable assignment or the assignment of a faulty UDS/SQL pubset declaration when the utility routine starts results in the utility routine aborting.

For details on the UDS/SQL declaration, please see section “Pubset declaration job variable”).

12.4 Automatic realm extension

Some utility routines automatically extend the realms of the processed database or the DBTTs in the DBDIR and DBCOM when required. For details, please see section “Automatic realm extension by means of utility routines”).

12.5 Outputting database information in a neutral format

The information-supplying utility routines BPSIA, BSTATUS, BPRECORD, and BOUTLOAD support data output in so-called CSV format (**c**haracter **s**eparated **v**alues). This format facilitates the further processing of the data in other system environments (e.g. in spreadsheet applications).

A semicolon is used to separate the individual values since commas are used in the data output.

12.5.1 Differences between CSV output and output to SYSLST

Optionally, data in CSV format may also be sent to SYSLST for output. The content of the output data in CSV format is similar to that of the SYSLST output. However, there are some differences in the format of the output data due to the fact that the data will not subsequently be used in paper form but will be further processed in other programs:

- All output in CSV format is stored in a file as part of a utility routine run.
- The first column contains an identifier that permits the line-by-line evaluation of the data.
- SYSLST output is restricted to the printing width of 132 characters per line. No such restriction applies to CSV output. Text information, e.g. headers, that has to be broken into several lines for SYSLST output is output in a single line in CSV format.
  Abbreviated text components are written in full in CSV format in order to achieve greater clarity.
- There are no page-oriented subheads in the CSV output.
- Repetitive explanatory texts from subtables are moved to additional headers.
- The header line of each CSV file contains the name of the utility routine, the corresponding utility routine's output format version, and the date and time of creation of the CSV output.

  The version changes if the CSV output changes with respect to the assignment to output cells or the semantic of output cells.
- Number notations which are represented in special formats in the SYSLST output for reasons of space (e.g. 25M or 14K in BSTATUS) are depicted in the correct digital form in the CSV output.
- Unlike in the SYSLST output, uppercase/lowercase notation is used for explanatory texts in the CSV output. Technical UDS/SQL terms and names declared in the database continue to be written in uppercase only.
- In the CSV output, BSTATUS also specifies the database name and the schema name.
- BPRECORD outputs record contents and table entries in Hex and Char format to the CSV file in compact form. In the Char format any hexadecimal character may occur. If the CSV file is transferred to a different system environment then the presence of individual characters which have special meanings may destroy the line structure of the CSV output.

  For example, on conversion to ISO8859-1 (ASCII), the EBCDIC character X'25' is converted to X'0A'. In Unix and Windows environments, X'0A' is interpreted as \n and therefore as a line feed. The EBCDIC character X'0D' remains unchanged on a switch from EBCDIC to ASCII. In a Windows environment it acts as an \r and generates a line feed. In a Unix environment, it returns to the start of the current line and may therefore result in elements of the output being overwritten.

  A simple way of avoiding such disruptive effects is to replace the characters in question with others while still in the BS2000 environment. This can be done, for example, in EDT with the following statements

  `@ON & CA X'25' T '?'` and

  `@ON & CA X'0D' T '?'`

  If the separator character occurs in the CHAR- output from BPRECORD, it is converted into X'00' in the CSV output in order to avoid any impairment to the continued higherlevel processing of the fields. If the separator character is required for further processing then this must be performed using the unchanged HEX output.

12.5.2 Controlling the CSV output with DISPLAY statements

You control the CSV format output using the IN CSV parameter in the DISPLAY statements available in the utility programs BPSIA, BSTATUS and BPRECORD (see the manual "Recovery, Information and Reorganization"):

```
DISPLAY [IN CSV [csv-filename]] ...
```

csv-filename

Name of the file that is to be output in CSV format.

Data output in CSV format is optionally additional to the output to SYSLST.

CSV format output is written to the file specified in the first DISPLAY statement. As a result, the *csv-filename* specification is mandatory in the first DISPLAY statement with CSV output.

The *csv-filename* specification can be omitted in subsequent DISPLAY statements with CSV output. If subsequent DISPLAY statements with CSV output contain a different *csv-filename* specification then a warning is issued. The output continues to be written to the originally specified file.

In general terms, the CSV output can be controlled for the individual DISPLAY statements. Internally, some of these DISPLAY statements are reorganized and combined in order to improve performance when the utility routine is run. In the case of complex DISPLAY statement combinations, this may result in more data being written in the CSV output than would be expected from a strict interpretation of the issued DISPLAY statements. This applies in case of BSTATUS to combinations of DISPLAY TABLE SET and DISPLAY TABLE OWNER. In BPRECORD, the control of CSV output applies to all output of a given type (PAGE, FPA, DBTT, CALC or DATA) for a realm.

If in such cases, you only want to generate the intended CSV format output, you can achieve this by performing multiple isolated utility routine runs.

12.5.3 CSV output in BOUTLOAD utility

The following statement allows you to output to CSV format in BOUTLOAD:

```
COPY-RECORD ..., CSV-OUTPUT = *YES
```

Data output in CSV format is created additionally to the output in the output file.

It is not obligatory to create the CSV output file. It is always created by the BOUTLOAD utility on a public disc (see the "Creation and Restructuring" manual).

12.5.4 Examples of usage

You can find examples on using CSV output in the selectable unit UDS-SQL-T as well as on the CD "UDS/SQL Tools and Drivers" that is shipped with UDS/SQL.

12.6 Job switches

In certain situations, the UDS/SQL utility routines set job switches or evaluate job switches set by the user:

Switch 29:	Evaluated by the UDS/SQL utility routines when logging is activated/deactivated and when errors occur in the logging file. If this switch was set by the user then the utility routine outputs console messages.
Switch 30:	If a WARNING occurs while the utility routine is executing, then the utility routine sets process switch 30. This continues to be set after the utility routine has terminated and can be evaluated by the user.
Switch 31:	If an ERROR or SYSTEM ERROR occurs while the utility routine is executing, then the utility routine sets process switch 31. This continues to be set after the utility routine has terminated and can be evaluated by the user.

Further information on the job switches is provided in the description of the BCHECK and BREORG utility routines in the "Creation and Restructuring" manual.

13 Using IQS

If the Interactive Query System IQS is used, it is the responsibility of the database administrator to ensure that every IQS user can work with the database without problems.

This means that the DB administrator must do the following:

- incorporate all object modules, load modules and procedures of IQS under one identification
- ensure that the BS2000 SORT subroutine module, the EDT subroutine module and the MFHSROUT module are available in the system tasklib
- assist the IQS user in eliminating errors.

13.1 Files, modules and procedures used by IQS

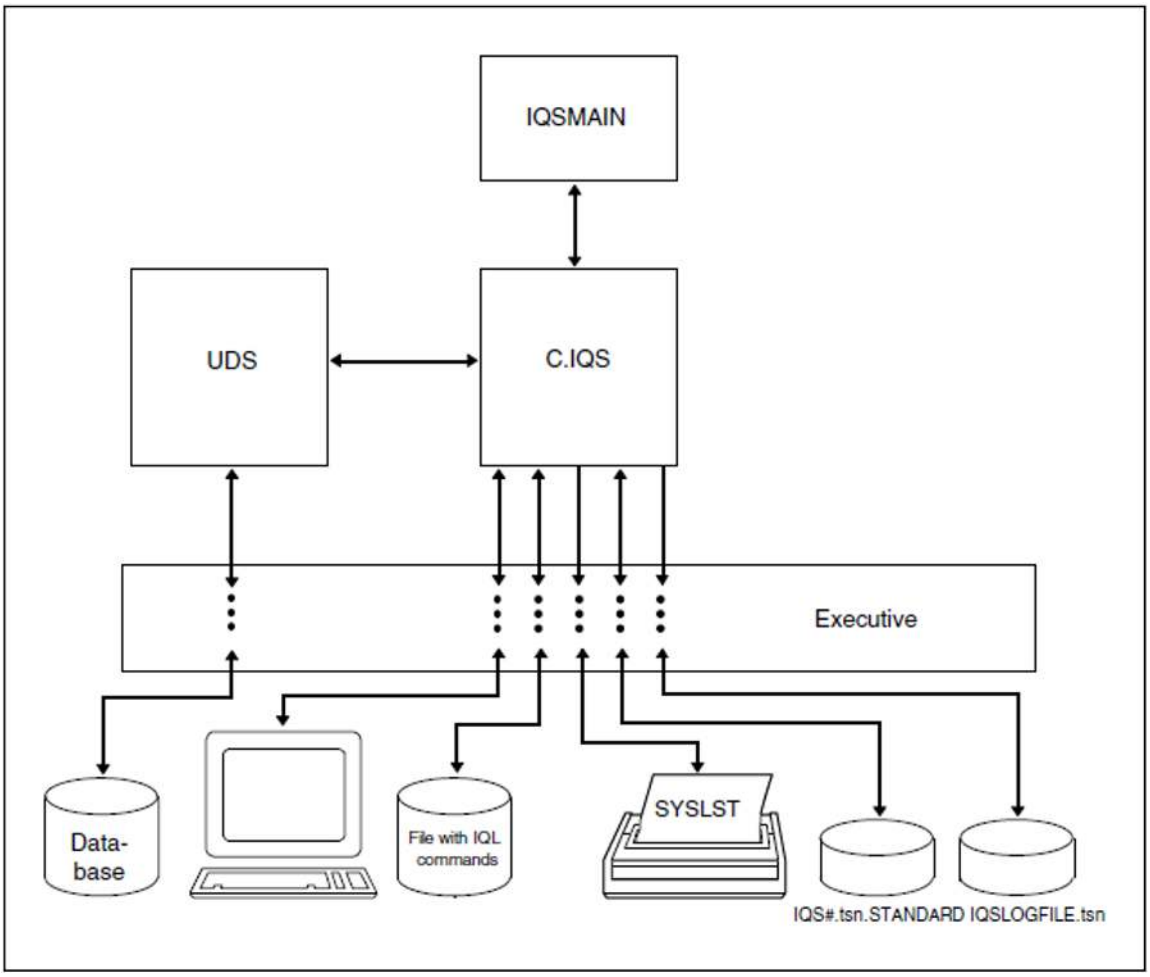


Figure 12: Interaction of IQS, UDS/SQL and operating system

C.IQS

IQS load module for independent DBH

C.L.IQS

IQS load module for linked-in DBH

IQSMAIN

IQL processor program, executable as shareable code in class 4 or class 5 memory (see the "Subsystem Management in BS2000" manual). IQSMAIN is loaded automatically by IQS from the module library SYSLNK.IQS.040. Its main memory requirement is approx. 188 Kbytes.

IQSGERM
IQSENGL

IQL language modules containing the IQL messages/error messages. The modules are reloaded from SYSLNK.IQS.040 and can be held in class 4 or class 5 memory.

IQS#.*tsn*.STANDARD

Temporary work file (user file) which is automatically created when the FIND command is used and deleted again when the IQS session terminates normally. If termination is abnormal, the file must be deleted with the DELETE-FILE command.

IQS.LOGFILE.*tsn*

Log file which receives IQS session logs.

SYSPRG.IQS.040

Library which contains the IQS load modules.

SYSLNK.IQS.04

Object module library which includes the IQL processor program IQSMAIN.

IQS.PRC.START

Start procedure for IQS. This procedure is contained in the library SYSPRC.IQS.040.

IQS dynamically loads the object modules it requires for UDS/SQL from a library which is assigned with the link name UDSOML via the ADD-FILE-LINK command.

```
/ADD-FILE-LINK LINK-NAME=UDSOML,FILE-NAME= ...,
```

The other connection modules of the independent DBH or the linked-in-DBH are dynamically loaded as with other application programs (see the “Application Programming” manual).

13.2 Diagnostic aids

IQS produces an extensive IQS log with BIB logging for the purposes of error diagnosis if SET LOG = BIB was used in the IQS session. This log can be printed out using the BWRLOG print utility (see "BWRLOG" in the "Interactive Query System IQS" manual), but this should be done only in the event of an error, since a large amount of paper is used.

If an IQS command has been wrongly processed, the following diagnostic documents should be generated:

- IQS log (SET LOG = BIB)
- DDL and SSL input file
- LMS directory of the SYSLNK.IQS.040

In the event of abnormal termination in addition: a dump

14 Using UDS-D

This chapter describes UDS-D, an add-on product for UDS/SQL that enables UDS/SQL databases to be processed on different systems. The chapter contains, in particular:

- a description of the product, its features, and notes on its use
- an overview of the options and general operation of UDS-D
- a description of the method used to ensure network-wide consistency
- supplementary information required for database operation with distributed databases.

14.1 Brief product description

UDS-D is an add-on product for UDS/SQL that enables the processing and administration of UDS/SQL databases in homogeneous BS2000 computer networks.
UDS-D reduces the load on a system by eliminating the fixed association of databases and applications with the same computer. In other words, data can be maintained on the system that is used most often, and other systems can then query or update this data as required. This makes all the data held in distributed databases accessible to both the central offices and the branches of a company, for example.

UDS-D provides unrestricted support for distributed transaction processing with networkwide deadlock recognition and handling using COBOL and CALL DML statements.

UDS-D is easy to use, since the distribution is not visible at the user interfaces. This means, on one hand, that the programmer need not know on which computer specific data is stored and, on the other, that application programs can be used without modifications on all computers on the network.

The main advantages of using distributed databases with UDS-D can thus be summarized as follows:

- Implementation of different organizational concepts:
  - Distributed databases can be used to map decentralized organizational structures. The data is processed at the location where it is generated (decentralization).
  - Isolated applications can be integrated into a single DB or DB/DC application across multiple systems.
- Despite decentralization, the entire data repository can be accessed directly, thus making it easy to support ad-hoc queries and reports, for example.
- System bottlenecks can be prevented by adding a further processor.
- Availability is increased in the case of system failures.

UDS/SQL can be used in combination with the distributed transaction monitor openUTM-D, both with and without UDS-D.

The following basic differences must be noted in the distribution concepts of UDS-D and openUTM-D:

UDS-D

- distributed databases
- communication per DML statement
- distribution not visible to application program
- multi-DB programs compatible with programs for distributed processing

openUTM-D

- distributed applications
- communication per transaction
- distribution visible to application program

Features of UDS-D

UDS-D is characterized by the following features:

- Access to multiple database configurations

  UDS-D enables application programs to access the databases of multiple mono- or multi-DB configurations. These DB configurations may be located on physically separate hosts that are connected via a network.

  The type of distribution used can be customized exclusively to suit the requirements of users and is not subject to any restrictions, but each database should, in principle, be physically located wherever it is most frequently processed. When databases of remote DB configurations are accessed, data is transported via the network, so the response times in such cases will primarily depend on the transmission facilities of the network. Database accesses via the network may therefore be much slower than those for the local DB configuration.

- No site-specific restrictions for application programs

  The user does not need to know how the databases are actually distributed, and only the names of the subschemas to be processed must be known to the application program. The associated DB configuration and the corresponding host system need not be known.
  Application programs can work with databases without knowing to which DB configuration they are connected. In other words, the application programs need not include any distribution-specific information.
  DML statements are sent to the DB configuration on the host where the addressed database is located by means of a distribution table.

- Cross-configuration consistency and deadlock resolution

  UDS-D guarantees the logical consistency of all databases involved throughout the network.

  If a transaction has updated the databases of at least one remote DB configuration, UDS-D will terminate that transaction only after ensuring that the updates are performed on the databases of all DB configurations involved or on none of the databases. This is achieved by UDS-D by splitting the process of terminating a transaction into two phases, i.e. by means of a two-phase commit. The first phase ensures that all updates of the transaction have been recorded safely, and the second phase then writes the updates to the databases only after the first phase has been successfully completed. This ensures that all updates can either be committed or rolled back at any time on terminating the transaction.

  Cross-configuration deadlocks are automatically detected and resolved.

14.2 Notes on using UDS-D

This section contains some notes on the aspects to be observed when designing, programming and restructuring databases which you want to reference via UDS-D connections in distributed transactions.

14.2.1 Design and programming

The response time behavior on accessing remote configurations essentially depends on the length and the amount of data to be transmitted. The following recommendations are made, taking this aspect into account:

- As far as possible, transactions should essentially contain accesses to local databases.
- The amount of data to be transmitted is reduced in COBOL DML programs if the programmer uses the smallest possible subschemas.
  Definitions such as SUBSCHEMA=SCHEMA should not be used!
- The amount of data to be transmitted for CALL DML programs is less than for COBOL-DML programs, since only the current record and not the entire RECORD-AREA is transmitted for CALL DML.

In the case of distributed databases, status code 141 (“Invalid subschema definition“) may be generated due to the following causes:

- The addressed subschema
  - is not contained in the local configuration and is not specified in the distribution table.
  - is specified in the distribution table, but is not present in the corresponding configuration.
  - is specified in the distribution table, but the appropriate configuration is unavailable, either because the computer cannot be accessed or because UDS-D mode has not been turned on in the running configuration.
  - is contained in the distribution table, but is locked, or the associated database or configuration is locked.
  - is not contained in the local configuration, and UDS-D mode was not started in the local configuration.
- The number of remote databases addressed by this transaction exceeds the value of PP DISDB.
- A configuration is temporarily inaccessible due to a networking problem. It is therefore advisable to repeat the transaction.

14.2.2 Reorganization

After reorganizing or regenerating a database, you must

- transport SSITAB modules to a remote CALL DML program.
- compile and link COBOL-DML programs. To do this, you will need information from the COSSD, which may be located on a remote host.

14.3 Working of UDS-D

The following diagram contains an example to show how the distribution of UDS/SQL databases can be achieved via a link using UDS-D.

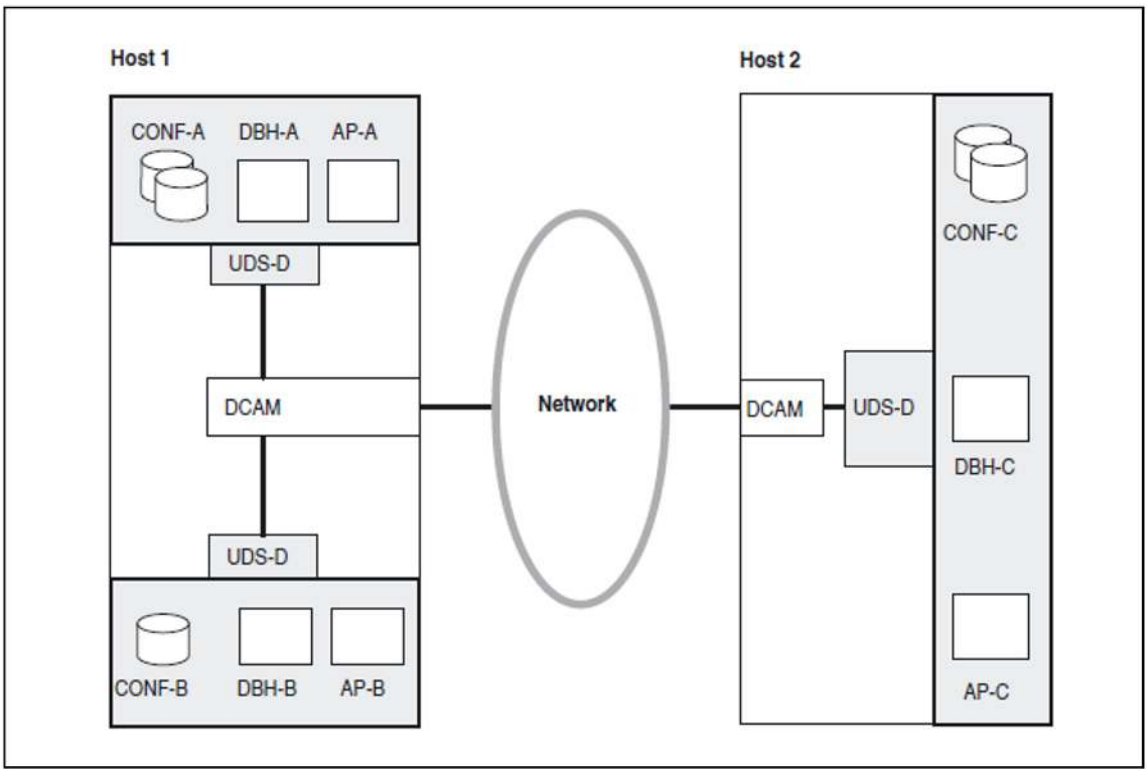


Figure 13: Distributed databases with UDS-D

An application program is connected to a configuration at startup with the commands:

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=configuration-name
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=...,ELEM=...)
```

This configuration is considered the **local** configuration in relation to the application program. In figure 13, the configuration CONF-A is local to the application program AP-A; configuration CONF-B is local to the application program AP-B, and so on.
All other configurations are **remote** configurations for the application program, i.e. the configurations CONF-B and CONF-C are remote for the application program AP-A.
In the reverse case, every application program that is connected to a configuration is considered local to that configuration; all other application program are remote. In other words, the application program AP-A is local to the configuration CONF-A, and the application programs AP-B and AP-C are remote.

14.3.1 The subtransaction concept

The subtransaction concept is an extension of the multi-DB transaction concept (see “Transaction in multi-DB operation” in the “Application Programming” manual).

A processing chain is a sequence of DML instructions to a single database within a transaction.

All processing chains in a distributed transaction that address the databases of **one** configuration constitute a **single** subtransaction.

Example

Consider the following configurations and databases:

CONF-A	CONF-B
DB-A1	DB-B1
DB-A2	

In this case, a transaction may contain the following subtransactions and processing chains:

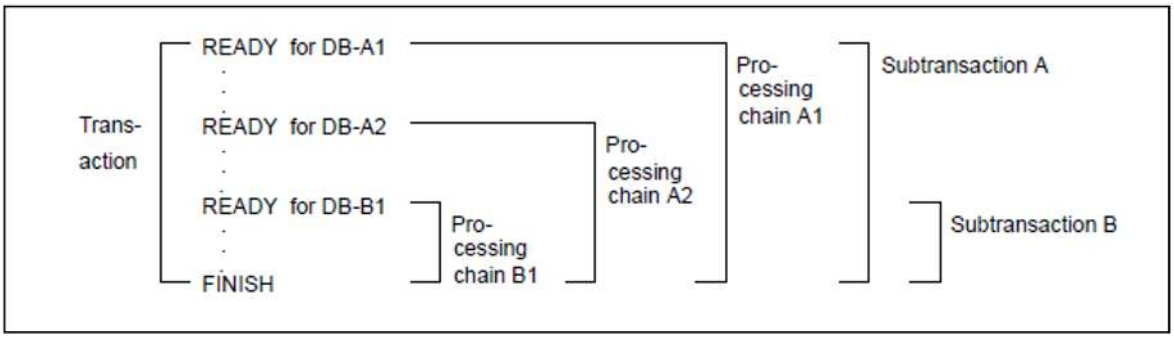


Figure 14: Transaction, subtransactions and processing chains for distributed databases with UDS-D

Figure 14 shows a **distributed transaction**. A transaction is considered to be distributed if it accesses at least one database in a remote configuration.

14.3.2 Primary and secondary subtransactions

All processing chains that address the databases of **one** DB configuration form a **single** subtransaction.

The subtransaction that addresses the local configuration is called the primary subtransaction, and all other subtransactions are called the secondary subtransactions (see example).
In other words, the primary subtransaction always runs in the local configuration, and secondary subtransactions always run in a remote configuration.

Example

Consider the following configurations and databases:

CONF-A	CONF-B	CONF-C
DB-A1	DB-B1	DB-C1
DB-A2		

In this case, the application program that triggers the sample transaction below is connected to the configuration CONF-A. Consequently, the following applies:

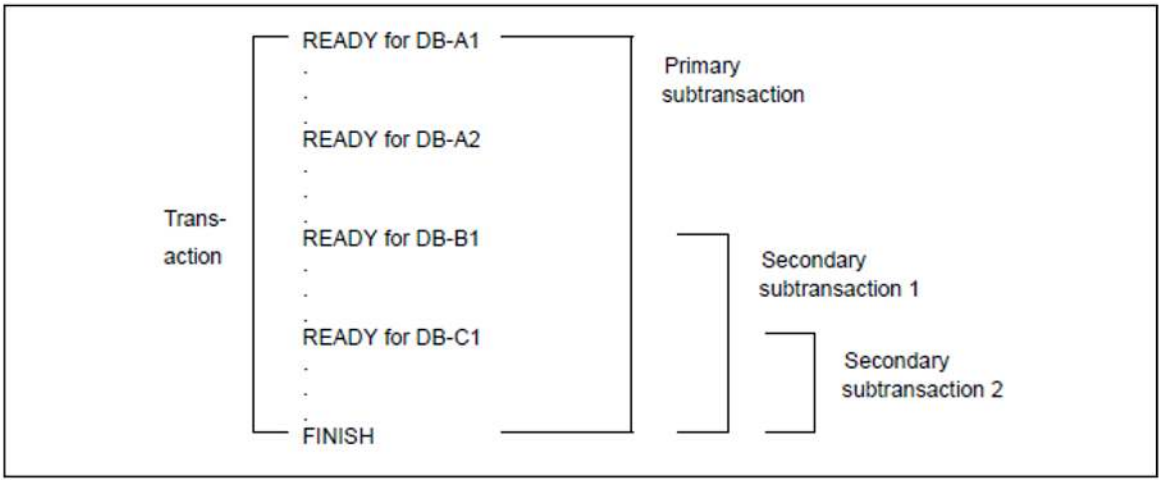


Figure 15: Primary and secondary subtransactions

> The first READY statement of a transaction opens the primary subtransaction. If the first READY statement of a transaction addresses a remote database, UDS-D generates a so-called dummy subtransaction as the primary subtransaction. As soon as a READY statement of the transaction addresses a local database, this dummy subtransaction becomes a real primary subtransaction.

14.3.3 Multi-DB programs for distributed processing

Multi-DB programs can be used for distributed processing without changes. A multi-DB program accesses databases that belong to one DB configuration. With UDS-D, a multi-DB program can also access databases belonging to multiple DB configurations, which in turn may be located on several different hosts.

Figure 16 shows a multi-DB program, with the distribution of databases depicted in grey. This distribution has no effect on the multi-DB program.

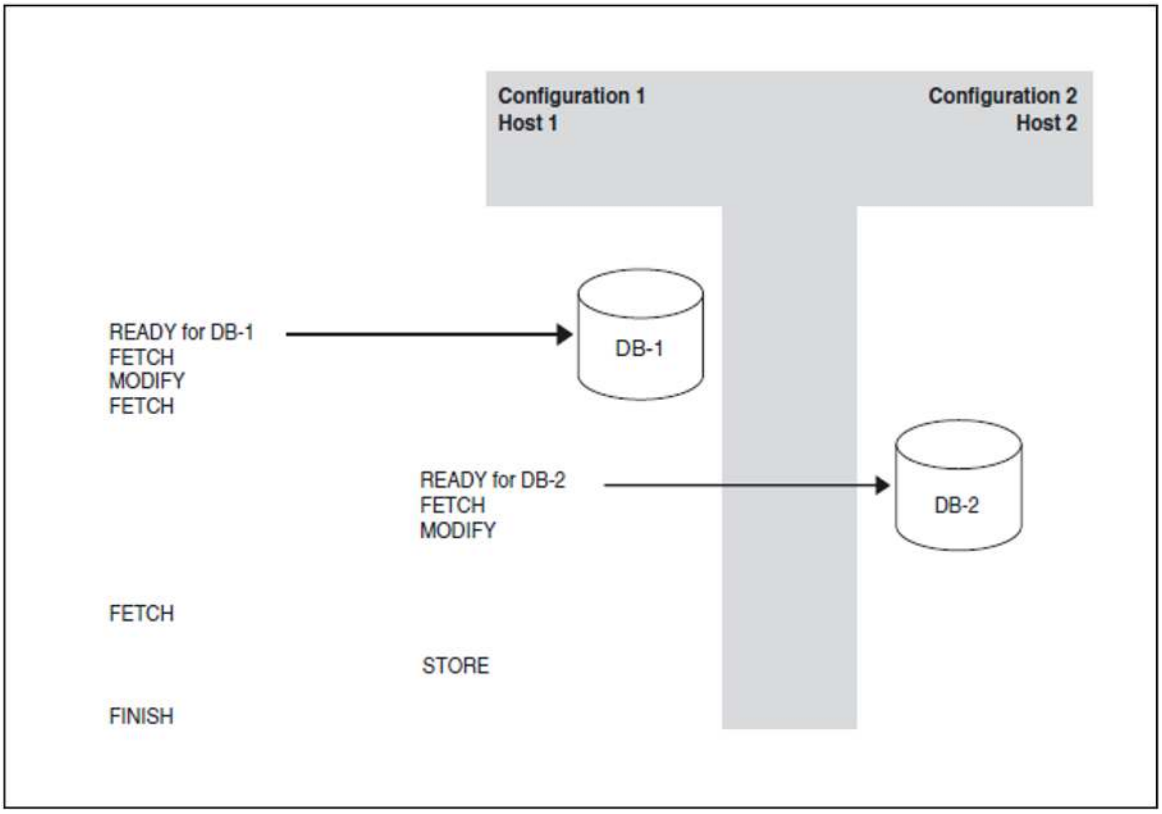


Figure 16: Multi-DB program for distributed processing

14.3.4 Communication between configurations

UDS-D is essentially implemented in a configuration by

- the UDS-D task (UDSCT)
- a distribution component in the application task (module name UDSNET).

Every configuration that is to communicate with remote configurations must be running UDS-D, i.e. must have the UDS-D task started in it.
The UDS-D task handles all the communication required to process DML statements received from remote application programs.

When UDS-D is used, the distribution component in the user task is loaded dynamically by UDS/SQL. The distribution component forwards the DML statements that come from a local application program and are to processed in a remote configuration. This occurs as follows:

When the DBH is started, the UDS-D task sets up a distribution table as specified (see section “The distribution table”). This table contains the following assignments:

- Subschema: Database
- Database: Configuration and
- Configuration: Host processor and password

When UDS-D mode is started, the distribution component in the user task uses this distribution table for each READY statement to determine where the DML statements for the processing chain involved are to be sent.

Communication between two configurations occurs by means of DCAM applications, which interact via logical connections. This applies even if the configurations are located in the same host.

Figure 17 shows the communication between two configurations, with one application program in one configuration using distributed processing.

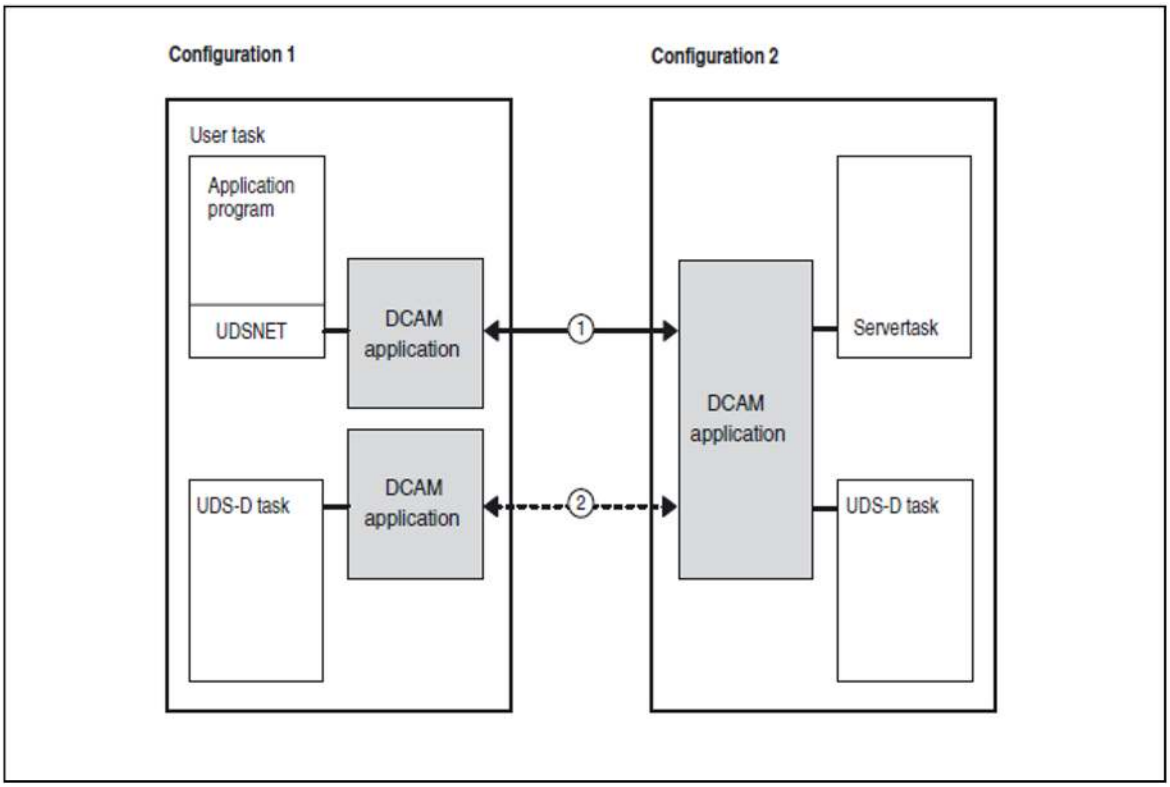


Figure 17: DCAM applications and logical connections for distributed databases with UDS-D

1. This logical connection is used to process DML statements. It is set up once per user task as soon as the user task accesses a remote configuration for the first time.
2. This logical connection is used for monitoring and for a warm start. It is set up once per configuration if required and is subsequently cleared when no longer needed.

14.4 Ensuring cross-configuration consistency with a two-phase commit

Whenever UDS-D terminates a distributed transaction, it always ensures that updates are either performed on the databases of all DB configurations involved or on none of the databases.

The application program terminates a transaction with a FINISH statement. This FINISH statement is automatically sent to all configurations in which subtransactions were opened.

In order to ensure that a distributed transaction that has updated at least one database of a remote DB configuration can be terminated consistently even in the event of an error, the process of terminating a transaction is split into two phases:

- In the first phase, UDS-D ensures that the updated pages of each subtransaction are securely logged in the RLOG file of the respective configuration.

  The successful completion of the first phase guarantees that all subtransactions can be terminated uniformly, i.e. either with FINISH or with FINISH WITH CANCEL.
- In second phase, UDS-D terminates all subtransactions with FINISH if the first phase was successful in all configurations. Alternatively, if the first phase was not successful in at least one configuration, all subtransactions are terminated with FINISH WITH CANCEL.

This method of approach is known as a “two-phase commit”.

If the logical connection is lost or a system failure occurs during the termination phase of a distributed transaction, UDS-D ensures that cross-configuration consistency, i.e. consistency between UDS/SQL and openUTM, is maintained (e.g. in the case of a session restart or within the framework of transaction monitoring; see section “Monitoring secondary subtransactions in the PTC state”).

The execution sequence of the two-phase commit depends on how a transaction is distributed:

- only via UDS-D (see section “Sequential flow of a two-phase commit for distribution via UDS-D”) or
- via UDS-D and openUTM (see section “Sequential flow of a two-phase commit in UDS-D for distribution via UDS-D and openUTM”).

Distribution via UDS-D

The two-phase commit is controlled by the distribution component in the user task. The primary subtransaction does not enter the PTC (prepared to commit) state. See also section “PTC state”.

Distribution via UDS-D and openUTM

The two-phase commit is controlled by openUTM. In this case, even the primary subtransaction enters the PTC state.

14.4.1 Sequential flow of a two-phase commit for distribution via UDS-D

The two-phase commit is initiated by the FINISH statement in the application program.

A distinction is made in the two-phase commit between subtransactions with and without an UPDATE flag. Every subtransaction initially has no UPDATE flag, regardless of the open mode (READY) of the database(s). It is only when an updating DML statement is successfully executed for a database that a subtransaction receives the UPDATE flag.

First phase of the two-phase commit

1. The distribution component in the user task sends
   - the FINISH statement to all secondary subtransactions without an UPDATE flag.
   - a PETA (preliminary end of transaction) statement to all secondary subtransactions with an UPDATE flag.

   The PETA statement creates a fail-safe log of the following information in the RLOG file of the respective configuration:
   - all pages that were updated by the respective secondary subtransaction,
   - the roll-back information, and
   - the names of all configurations involved (in case a warm start is required).

   In the steps that follow, references to “secondary subtransactions” essentially mean “secondary subtransactions with an UPDATE flag”, since the remaining secondary subtransactions have already been terminated.
2. The secondary subtransactions execute the PETA statement and report to the primary subtransaction whether or not the PETA statement could be executed without errors.
3. If the primary subtransaction has not received a message from every secondary subtransaction, it checks whether the execution of the PETA statement was successful for all secondary subtransactions.

   Step 4a. or 4c. is then performed accordingly.

   > If the primary subtransaction has not received a message from all secondary subtransactions before a time interval that depends on PP CHCKTIME expires, the primary subtransaction is terminated with FINISH WITH CANCEL, and the secondary subtransactions are also instructed to terminate with FINISH WITH CANCEL.
   > The application program is assigned status code 122.

Second phase of the two-phase commit

4a.

If all secondary subtransactions have executed the PETA statement successfully, the primary subtransaction is terminated by committing the updates (with FINISH). If the FINISH on the primary subtransactions succeeds, step 4b. is executed.
Otherwise, i.e. if the FINISH on the primary subtransaction could not be completed successfully, the secondary subtransactions are rolled back.

4b.

All secondary subtransactions are instructed to terminate with FINISH.

4c.

If all secondary subtransactions could not successfully execute the PETA statement, the primary subtransaction is terminated by rolling back the updates (FINISH WITH CANCEL), and the secondary subtransactions are also instructed to terminate with FINISH WITH CANCEL.

14.4.2 Sequential flow of a two-phase commit in UDS-D for distribution via UDS-D and openUTM

First phase of the two-phase commit

1. openUTM sends a PETA statement to UDS/SQL. The distribution component in the user task sends
   - a FINISH statement to all secondary subtransactions without an UPDATE flag and
   - the PETA statement to all secondary subtransactions with an UPDATE flag.

   The PETA statement causes a fail-safe log of the following information to be created in the RLOG file of the respective configuration:
   - all pages that were updated by the respective secondary subtransaction,
   - the roll-back information, and
   - the names of all configurations involved (in case a warm start is required).

   In the steps that follow, references to “secondary subtransactions” essentially mean “secondary subtransactions with an UPDATE flag”, since the remaining secondary subtransactions have already been terminated.
2. The secondary subtransactions execute the PETA statement and report to the distribution component in the user task whether or not the PETA statement could be executed without errors.
3. If the distribution component in the user task has not received a message from every secondary subtransaction, it checks whether the execution of the PETA statement was successful for all secondary subtransactions.

   Step 4a. or 4c. is then performed accordingly.

   > If the distribution component in the user task has not received a message from all subtransactions before a time interval that depends on PP CHCKTIME expires, all subtransactions are instructed to terminate with FINISH WITH CANCEL.
   > openUTM receives the return code "Transaction rolled back".

4. 4a.

   If all subtransactions have executed the PETA statement successfully, the PETA statement is issued for the primary subtransaction as well. If this statement executes successfully, the entire DB transaction is placed in the PTC (prepared to commit) state; see the section “PTC state”. The PTC state is reported to openUTM, and control is handed over so that openUTM can initiate the second phase.
   If the PETA statement could not be executed successfully, step 4b. is performed.

Second phase of the two-phase commit

4b.

All secondary subtransactions are instructed to terminate with FINISH WITH CANCEL. openUTM receives the return code "Transaction rolled back".

4c.

If all secondary subtransactions could not execute the PETA statement successfully, all subtransactions, including the primary subtransaction, are instructed to terminate by rolling back the updates (FINISH WITH CANCEL). openUTM receives the return code "Transaction rolled back".

5. openUTM harmonizes with other applications how the transaction is to be terminated and causes UDS/SQL to commit or roll back the updates.

   The termination of individual subtransaction is controlled by the distribution component in the user task.

14.4.3 PTC state

A subtransaction is said to be in the PTC (prepared to commit) state if it has successfully executed the PETA statement, but has not yet received an instruction to terminate. In this state, no decision has been made as to whether the entire transaction, i.e. including the subtransaction itself, will be terminated with FINISH or FINISH WITH CANCEL.

In the case of a distribution via UDS-D, only secondary subtransactions with an UPDATE flag enter the PTC state. For a distribution via UDS-D and openUTM, the primary subtransaction may also enter the PTC state if it or at least one of its secondary subtransactions has the UPDATE flag.

The following diagrams illustrate the actions of the individual subtransactions and the PTC state

- when the two-phase commit leads to FINISH for a transaction distributed via UDS-D
  (see figure 18).
- when the two-phase commit leads to a rollback for a transaction distributed via UDS-D
  (see figure 19).
- when the two-phase commit leads to FINISH for a transaction distributed via UDS-D and openUTM
  (see figure 20).
- when the two-phase commit leads to a rollback for a transaction distributed via UDS-D and openUTM
  (see figure 21).

The following abbreviations are used in the figures below:

PTT: Primary subtransaction

STT: Secondary subtransaction

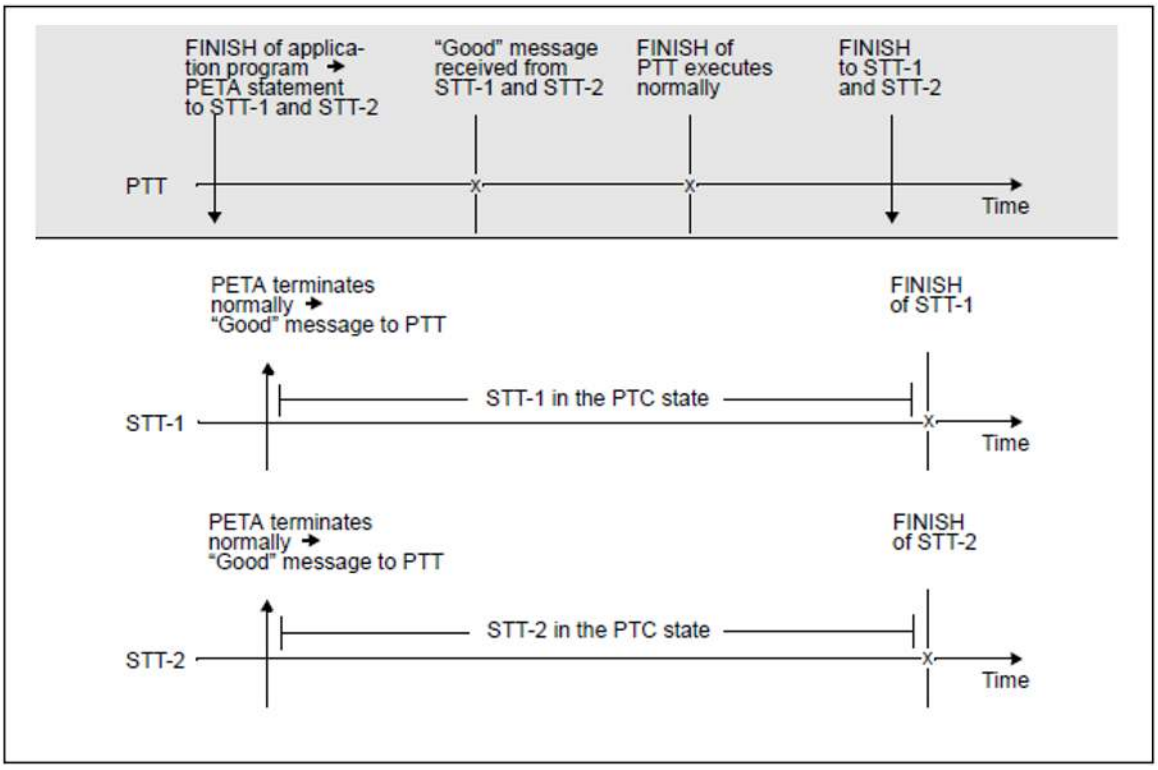


Figure 18: Chronological sequence of a two-phase commit with FINISH using UDS-D

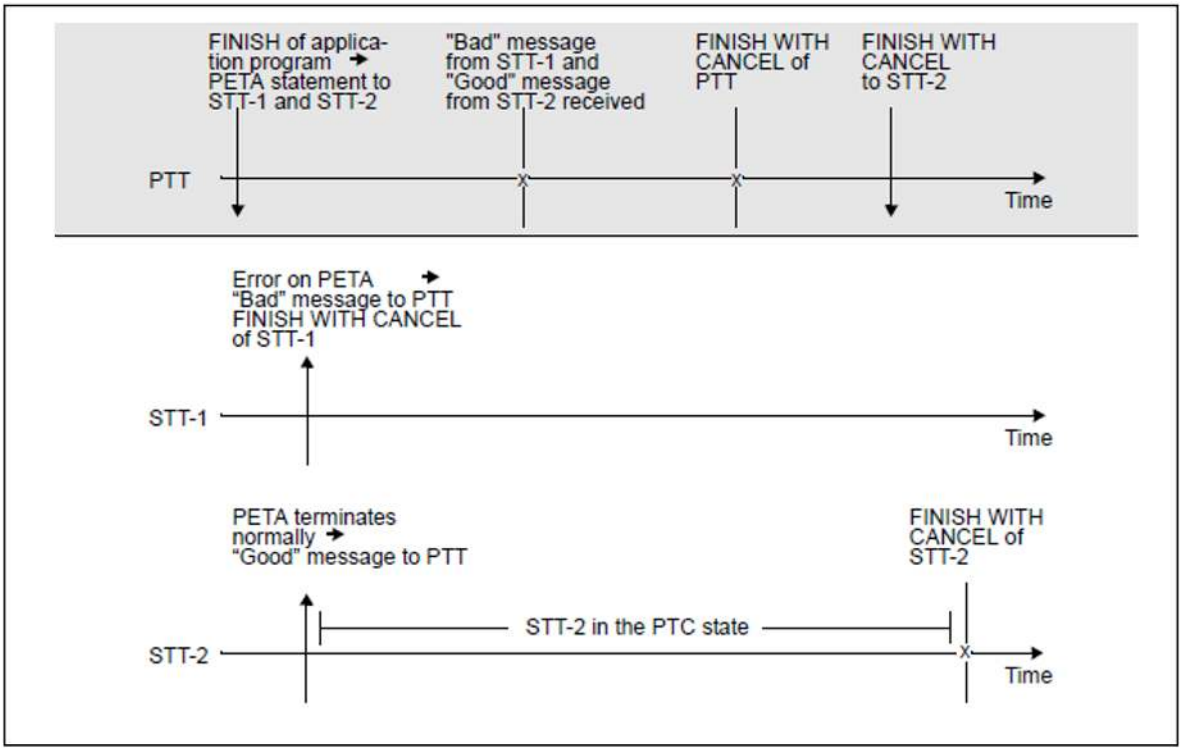


Figure 19: Chronological sequence of an unsuccessful two-phase commit using UDS-D

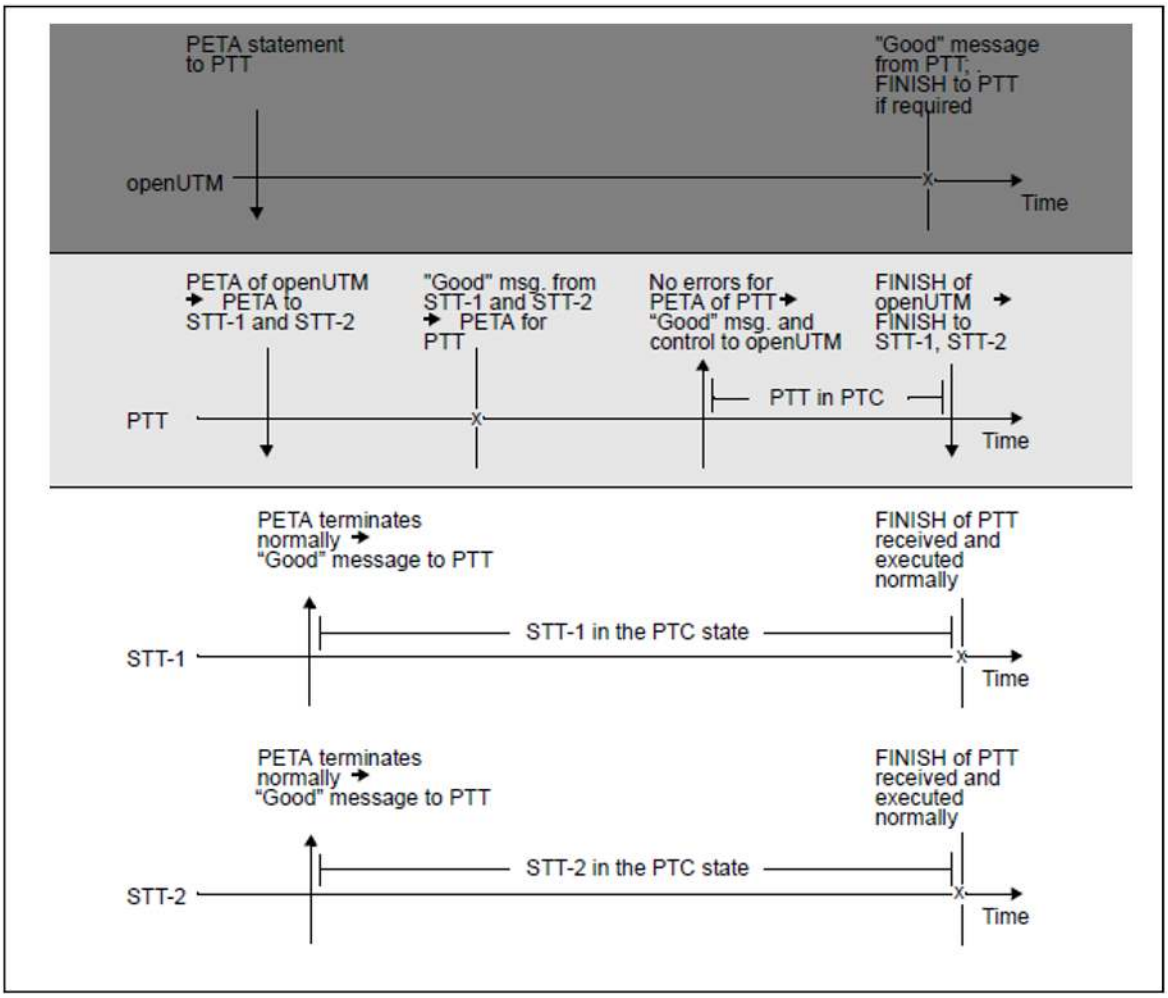


Figure 20: Chronological sequence of a two-phase commit with FINISH for distribution via UDS-D and openUTM

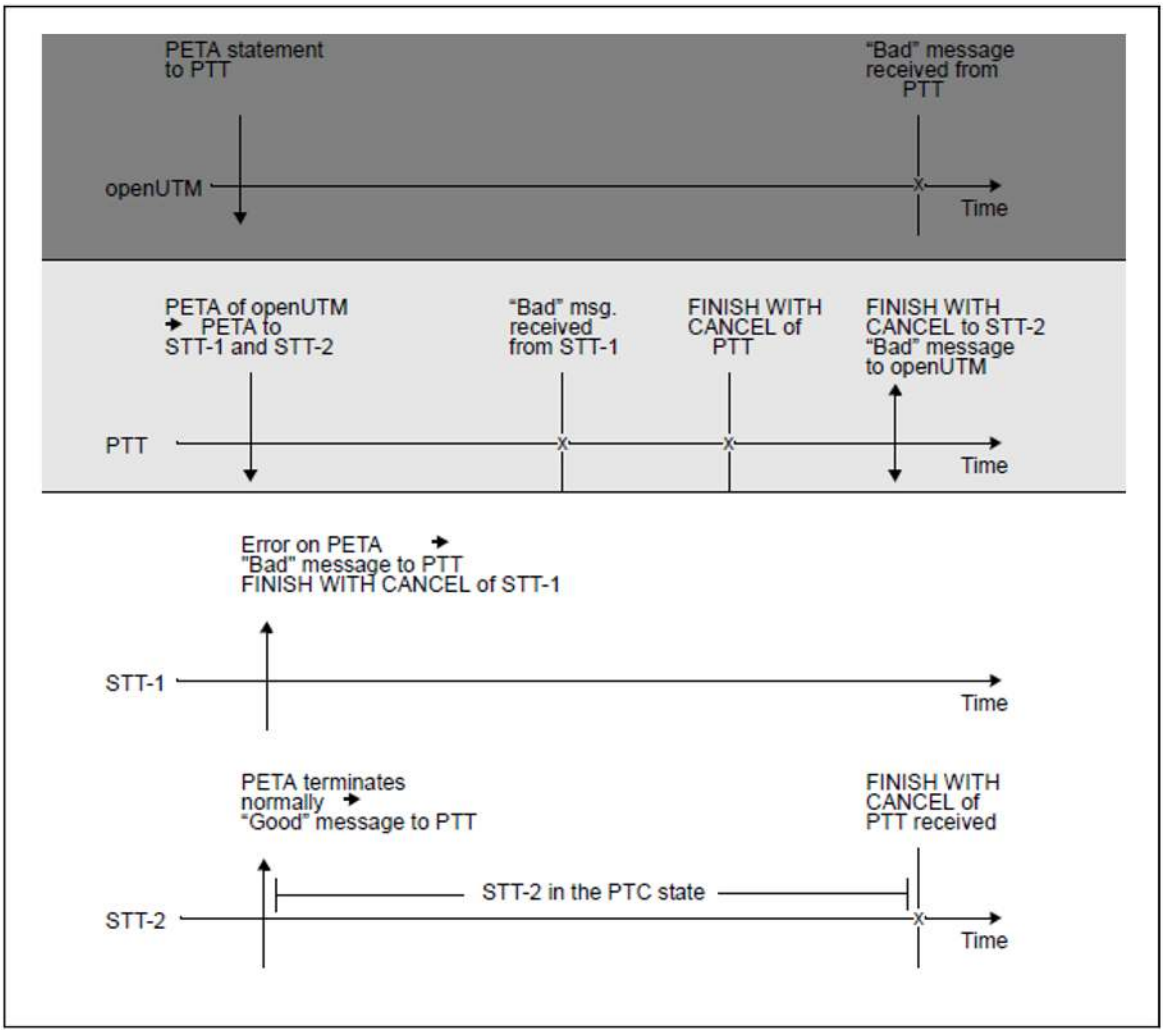


Figure 21: Chronological sequence of an unsuccessful two-phase commit for distribution via UDS-D and openUTM

14.4.4 Terminating the PTC state

In some cases, it may not be possible to terminate transactions in the PTC state, e.g. due to a failure in a partner system or transmission line. If the problem cannot be solved within an acceptable period, UDS-D offers the database administrator a facility to intervene in order to resume database operation. He or she can

- use the DBH load parameter PP PTCSYNCH to control whether secondary subtransactions or openUTM transactions in the PTC state should terminate with FINISH or with
  FINISH WITH CANCEL after a time interval defined by PP CHCKTIME. The value of the DBH load parameter PP PTCSYNCH can also be modified by the database administrator during a session by means of the DAL command MODIFY.
- use the DAL commands ABORT,OPTION=PTC or COMMIT to terminate secondary subtransactions or openUTM transactions with FINISH or with FINISH WITH CANCEL “manually”.

CAUTION!

When the PTC state is terminated by the database administrator, crossconfiguration consistency or the UDS/SQL and openUTM consistency will no longer be guaranteed as soon as

- the DBH load parameter PP PTCSYNCH is not set to (WAIT,WAIT) or
- the database administrator intervenes with the DAL command ABORT,OPTION=PTC or COMMIT.

When secondary subtransactions or openUTM transactions in the PTC state are terminated by the database administrator, updates of the same transaction may be completed in the databases of one DB configuration, but not in those of another.
UDS/SQL informs the database administrator how the transaction is distributed so that he or she can take steps to limit or eliminate the inconsistency.
The consistency within a DB configuration is maintained in any case.

14.5 Database operation with UDS-D

This section explains what you will need to ensure smooth operations with distributed databases. It describes the following:

- the UDS-D-specific parts of the DBH
- handling of a COBOL-DML or CALL DML statement that processes a remote subschema
- configuration-based password protection
- the distribution table
- time-controlled connection and transaction monitoring
- starting and terminating UDS-D mode
- effects of the PTC state
- administration of multiple configurations
- notes on modifying the DB configuration

14.5.1 The DBH when using UDS-D

Distributed databases can be processed in combination with UDS-D only by the independent DBH.

When UDS-D is used, the independent DBH consists of the following tasks:

- the master task
- one or more server tasks
- the UDS-D task

The **master task** and the **server tasks** are described in detail in section “How the independent DBH works”).

The **UDS-D task** (UDSCT) is loaded by the master task as an ENTER job in accordance with the DBH load parameter PP DISTRIBUTION.

The UDS-D task sets up its own specific tables and memory areas, e.g. the distribution pool with the distribution table and the transfer pool to accept BIBs. It receives DML statements from remote application programs and passes them on to the server task.
The UDS-D task continuously monitors the logical connections to remote application programs (see section “Monitoring the logical connection to remote application programs”).

The **transfer pool** performs a similar function for DML statements of remote application programs as that of the **communication pool** for DML statements of local application programs. It contains

- the transaction ID
- the DML statement with its parameters (BIB)
- the data to be transmitted (BIB)

The **common pool** is a memory area that can be accessed by all DBH requests. It is described in detail in section “How the independent DBH works”.

The **distribution pool** contains the **distribution table** and UDS-D-specific system tables.

The distribution table is used by the distribution component in the user task to determine whether the subschema to be processed is local or remote.

The figure 22 illustrates the independent DBH when using UDS-D. It shows the main communication channels that are used when a remote application program communicates with the DBH.

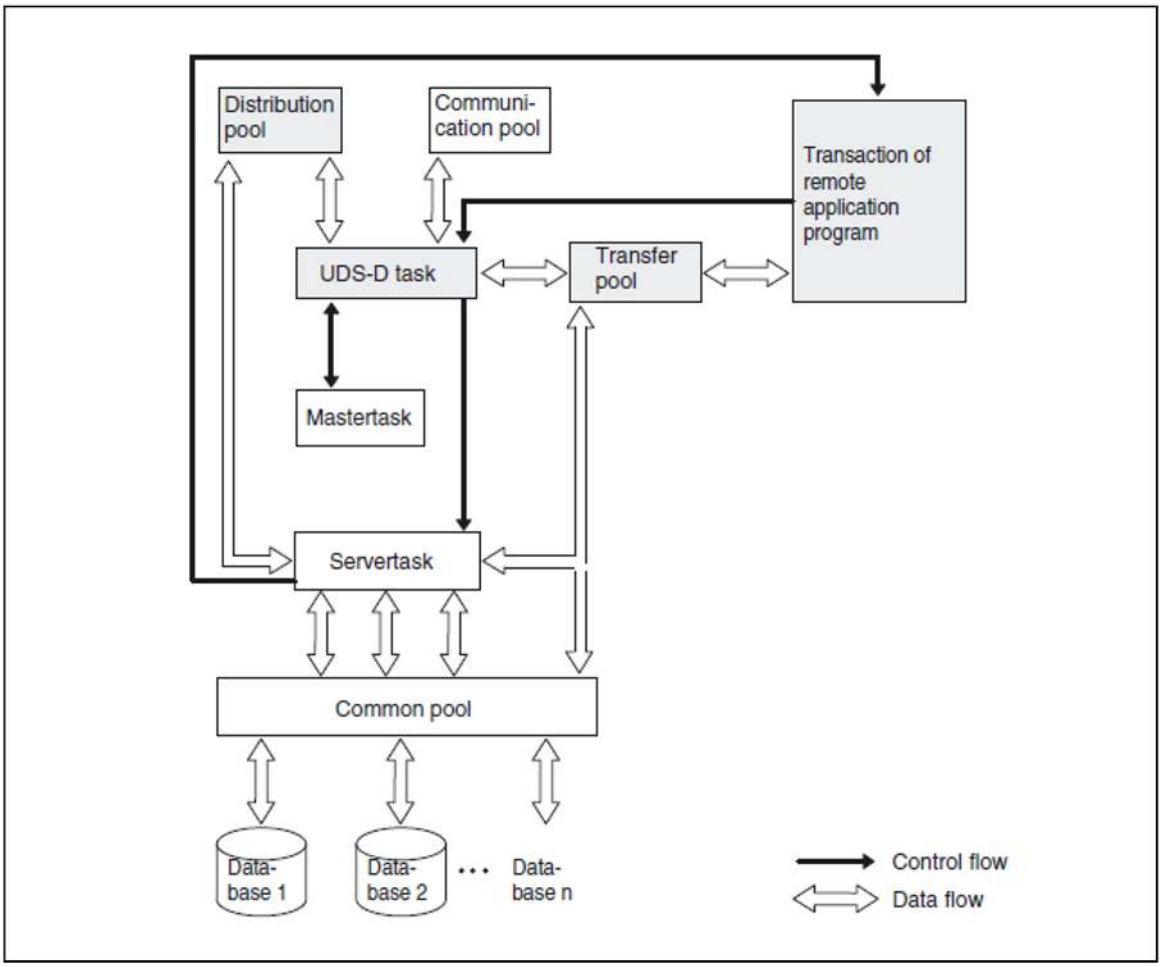


Figure 22: The independent DBH when using UDS-D

Handling of a COBOL DML or CALL DML statement for a remote subschema

The figure 23 shows how a COBOL DML or CALL DML statement of the application program is handled in cases where a remote subschema is to be processed.

Note that if a CALL DML statement is involved, a BIB is created to enable further processing of the request (see “Processing of a CALL DML statement by the independent DBH” in chapter "How the independent DBH works"). This BIB serves as the input for the distribution component UDSNET.

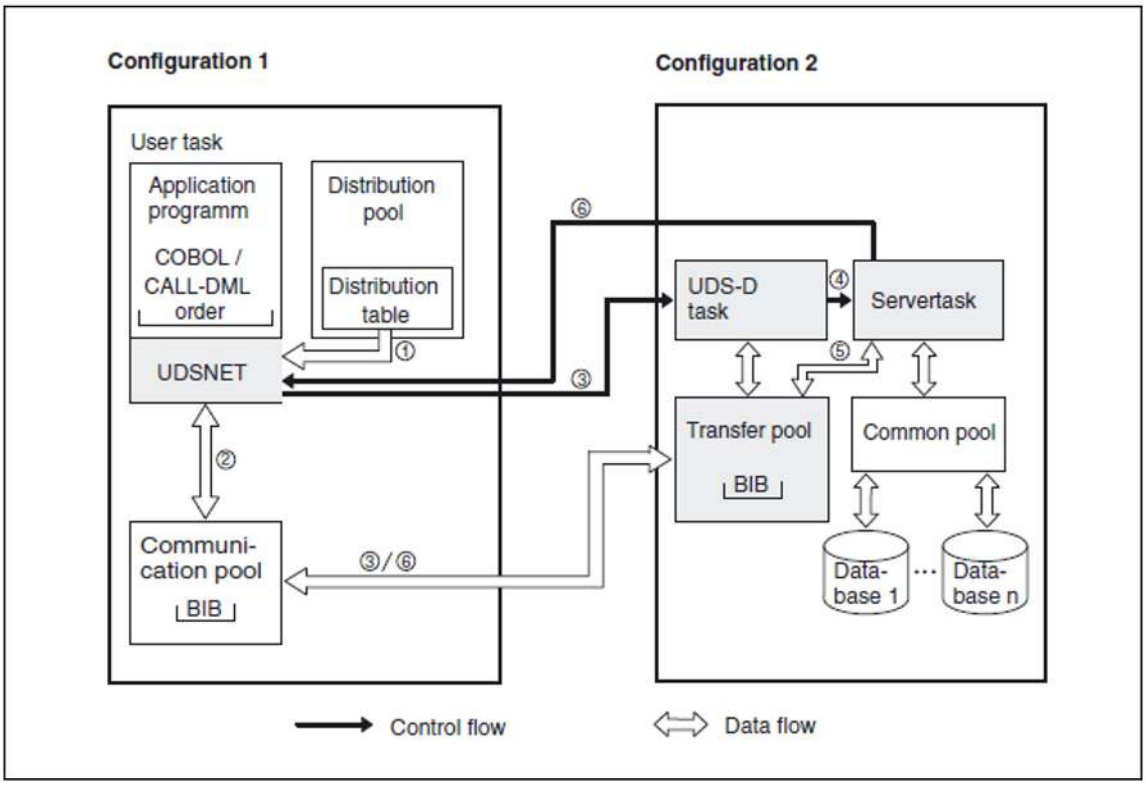


Figure 23: Handling of a COBOL-DML/CALL DML statement that processes a remote subschema

1. On receiving the appropriate READY statement, the connection module determines the location of the remote configuration via the distribution table.
2. The BIB that is to process a remote subschema is first transferred by the user task to the communication pool.
3. The BIB is then sent from the communication pool via the logical connection to the transfer pool. This awakens the remote UDS-D task. The user task itself waits for the arrival of the processed BIB.
4. The BIB is assigned to a currently available server task for processing.
5. The server task processes the BIB.
6. The BIB is then returned from the transfer pool via the logical connection to the communication pool. This terminates the wait state of the user task.

14.5.2 Configuration-based password protection

Concept

Every configuration can be protected against unauthorized access from the user tasks of other configurations by means of a configuration-based password. When a connection is set up, the password assigned to the local configuration serves as a lockcode, and those assigned to the remote configurations are used as keycodes (originally called locks and keys; see also the openUTM “Access control via openUTM”).

Whenever a request to set up a connection is initiated in a user task, the appropriate keycode password is supplied by UDS-D with the request. When the request arrives in the remote configuration, this keycode is compared in the UDS-D task with the corresponding lockcode. If the lockcode and keycode do not match, the request to establish the connection is rejected by the UDS-D task. The database request of the application program that initiated the connection setup is aborted, and any transaction that was already opened is rolled back.

Assigning passwords

Configuration-based passwords can be assigned in the input file for the distribution tables and by using the DAL command &PWD DISTRIBUTION (see section “The distribution table” and “Assigning and changing a password (&PWD DISTRIBUTION)”).

Example

Two UDS/SQL applications are running concurrently, one using a test configuration, and the other with a production configuration. In order to facilitate the incorporation of the test application for production use at a later stage, there are not differences between the two configurations with respect to the privacy information and the subschema and database names.

This approach can be supported by protecting the production configuration from the test configuration by means of a configuration-bases password. If the configuration and processor name of the production application is inadvertently entered in the distribution table of the test application, the password will prevent the test application from accessing the production database(s) in any case.

14.5.3 The distribution table

UDS-D enables an application program to access the databases of multiple DB configurations. The application program is only aware of the subschema name, so UDS-D must know whether the subschema is located in a local or remote DB configuration.

This information is made available by means of an input file, which is created by the database administrator for each DB configuration in which distributed transactions are to be started. This input file is used by UDS-D to create the so-called distribution table. This input file and the distribution table created from it contain the following assignments:

- Subschema: Database
- Database: Configuration and
- Configuration: Host processor and password

The distribution table is placed by UDS-D in the distribution pool so that the distribution component in the user task can access it during the session. The distribution table may also include local databases and local subschemas, so the same distribution tables can be provided for all DB configurations. If the distribution tables are large, it is advisable to place the local databases and subschemas at the start of the input file to ensure good response times for local transactions.
Databases and subschemas that are not described in the distribution table are treated as local.
Remote databases and subschemas that are not described in the distribution table cannot be accessed by a local application program.

14.5.3.1 Structure of the input file for the distribution table

The input file is a SAM or ISAM file. It contains database and subschema records, which are structured as follows:

- Database records:

  DB'BLANK'*dbname*'BLANK'*confname*'BLANK'*processor-name*['BLANK'*password*]
- Subschema records:

  SS'BLANK'*subschema-name*'BLANK'*dbname*

Every record must begin in a new line and may be up to 160 characters in length. Leading blanks are ignored by UDS-D.
If the *dbname*, *confname*, *processor-name* and *subschema-name* are to be initially treated as **locked**, they must be identified by a **preceding minus sign**. These databases and subschemas will then be accessible to the application program only when the lock has been removed with the DAL command &UNLOCK DISTRIBUTION.

Database and subschema records that begin with an asterisk (*) are treated as comments and are skipped when creating the distribution table. This enables the database administrator to create an input file with a clean layout.

UDS-D checks the syntax and ensures that names are assigned uniquely. Only the following assignments are allowed:

- *subschema-name* to only one *dbname*
- *dbname* to only one *confname*/*processor-name*

If the same name occurs again, UDS-D ignores the corresponding database or subschema record and issues a message.

In the case of a multi-DB configuration, many DB records will have the same entry for *confname*/*processor-name*. If a password is to be assigned to such a configuration, the password only needs to be specified with one of the associated DB records.

If different passwords for a configuration are specified in different DB records, the first password specified in the sequence of DB records applies. The other conflicting passwords specified with the DB records belonging to the same configuration are rejected with an appropriate message.

Database records

Database records specify the DB configuration to which a database belongs, possibly the password for that configuration, and the host on which the configuration is located.

The following naming conventions apply:

processor-name

The name of the host processor may be up to eight characters in length.

dbname

Database names in the distribution table must be unique throughout the network. See also table 5 in chapter "SDF syntax representation".

confname

The first eight characters of the *configuration-name*, which must be unique in the first seven characters. The eighth character must not be a “@”. Note that zeros at the end of *confname* are not considered significant characters, i.e. no distinction is made between *confname* ABC and *confname* ABC0.

password

The cross-configuration password can have a maximum length of eight characters.

Subschema records

Subschema records specify the database to which the subschema belongs.

The following naming conventions apply:

subschema-name

This name must be unique in the first six characters within the distribution table throughout the network. See also table 5 in chapter "SDF syntax representation".
Only user subschemas may be specified. The PRIVACY-AND-IQF subschema and COMPILER subschema are not permitted in the distribution table.

14.5.3.2 Reading the input file

The DBH reads the name of the input file for a distribution table with the DBH load parameter PP DISTABLE (see “Defining the input file for the distribution table (DISTABLE)” in chapter "DBH load parameters").

14.5.3.3 Structure of the distribution table

UDS-D creates a distribution table based on the contents of the input file (see figure 24).

The following entries are differentiated in the distribution table:

- NODE/CONF entry
  contains a processor name and the first seven characters of a configuration name, possibly padded with zeros and the eighth character “@”
- DB entry
  contains a database name
- SS entry
  contains the first six characters of a subschema name

The SS entry is linked to the associated DB entry, and the DB entry is linked to the associated NODE/CONF entry.

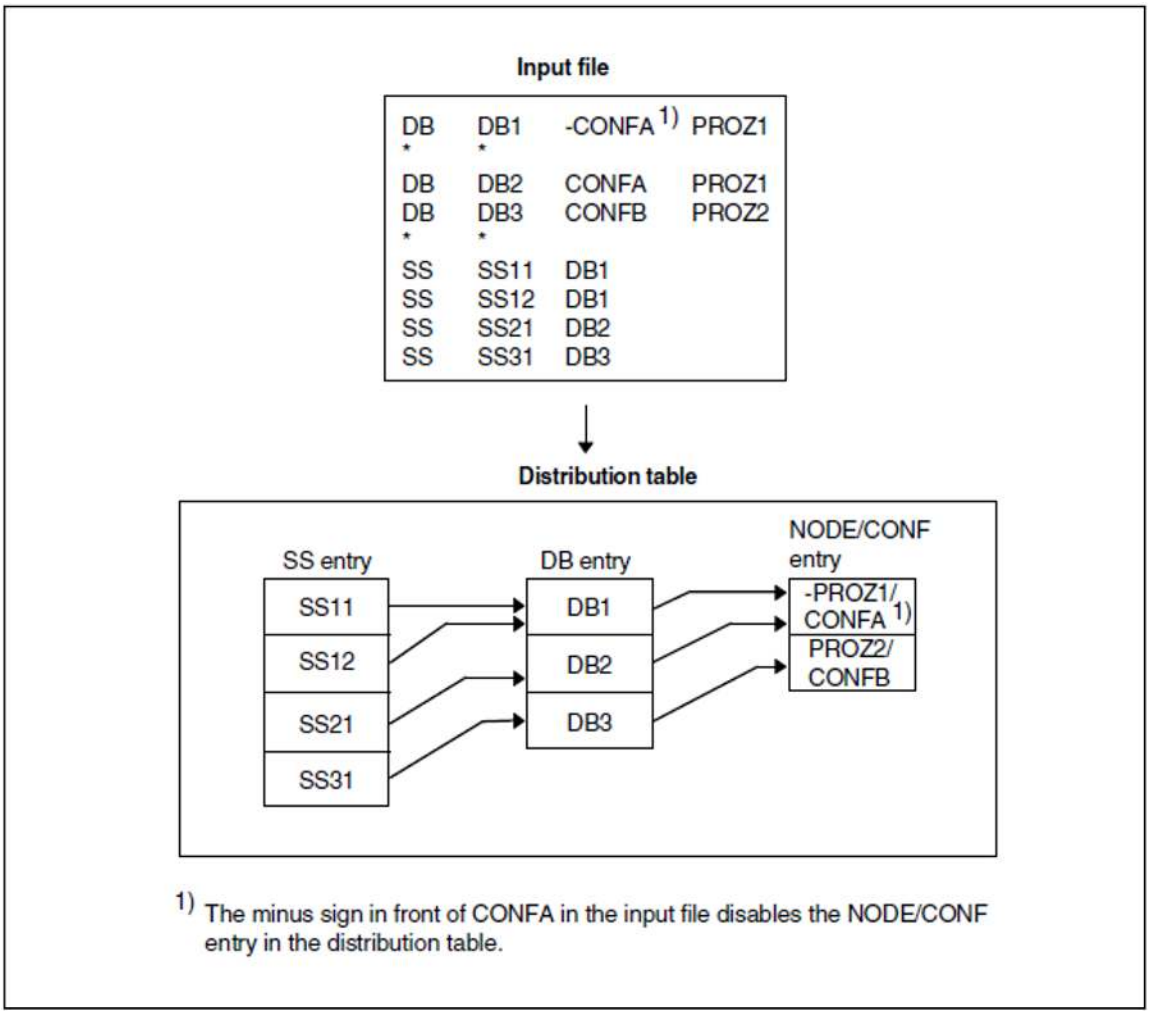


Figure 24: Structure of the distribution table created from an input file

The arrows in the distribution table point from the SS entry to the associated DB entry and from the DB entry to the associated NODE/CONF entry. This chain linking the SS entry to the DB entry and the NODE/CONF entry must be complete in order to enable the distribution component in the user task to send a DML statement to the desired subschema of a remote configuration. Otherwise, the application program will receive status code 141: “Invalid subschema definition”.

14.5.3.4 Overview of DAL commands for distribution tables

If required, the database administrator can use DAL commands to modify the local distribution table so that a complete chain remains intact or a new chain is created. The DAL commands are described in section “The Database Administrator Language DAL”.

DAL command	Meaning
&ADD DISTRIBUTION...	Add new entries to the distribution table
&CHANGE DISTRIBUTION...	Assign a configuration to another host, i.e. change the NODE/CONF entry
&DISPLAY DISTRIBUTION...	Display the distribution table on the screen
&DROP DISTRIBUTION...	Delete entries
&LOCK DISTRIBUTION...	Lock entries in the distribution table
&PWD DISTRIBUTION...	Assign or modify a configuration-based password
&SAVE DISTRIBUTION...	Save the distribution table in a SAM or ISAM file
&UNLOCK DISTRIBUTION...	Remove lock on entries

Table 32: Overview of DAL commands for distribution tables

- Changes in distribution tables do not take effect until the next READY statement.
- A database administrator who is not administering multiple DB configuration centrally (see section “Central administration”) is only allowed to modify the distribution table of his/her own DBH.
- It is not possible for different DBHs to synchronize the contents of their distribution tables.
- Changes in distribution tables have no effect on the actual location of a DB configuration.
- The output of the distribution table is restricted to 32 KB. If the distribution table is larger than this then it is not output in full and you are advised of this by a message.

14.5.4 Time-controlled monitoring of connections and transactions

UDS-D monitors operations with distributed transactions by means of a timer. The time interval to be selected for the timer is based on the DBH load parameter PP CHCKTIME (see “Monitoring connections and transactions (CHCKTIME)” in chapter "DBH load parameters").

UDS-D monitors

- the logical connections to remote application programs
- secondary subtransactions and checks whether the associated primary subtransactions still exist
- secondary subtransactions in the PTC state
- primary subtransactions that were terminated by the database administrator.

14.5.4.1 Monitoring the logical connection to remote application programs

UDS-D prevents a partner system from interpreting a long delay in the processing of a DML statement as a line failure or a failure in the corresponding partner system.

When the UDS-D task receives a DML statement from a remote application program, it sends time acknowledgments back to the remote application program. These time acknowledgments indicate that the DML statement is still being processed by the server task (see figure 25). The UDS-D task sends the time acknowledgments to UDSNET, the distribution component in the user task that handles communication requirements.
The time acknowledgments are sent at periodic intervals, which depend on the DBH load parameter PP CHCKTIME.

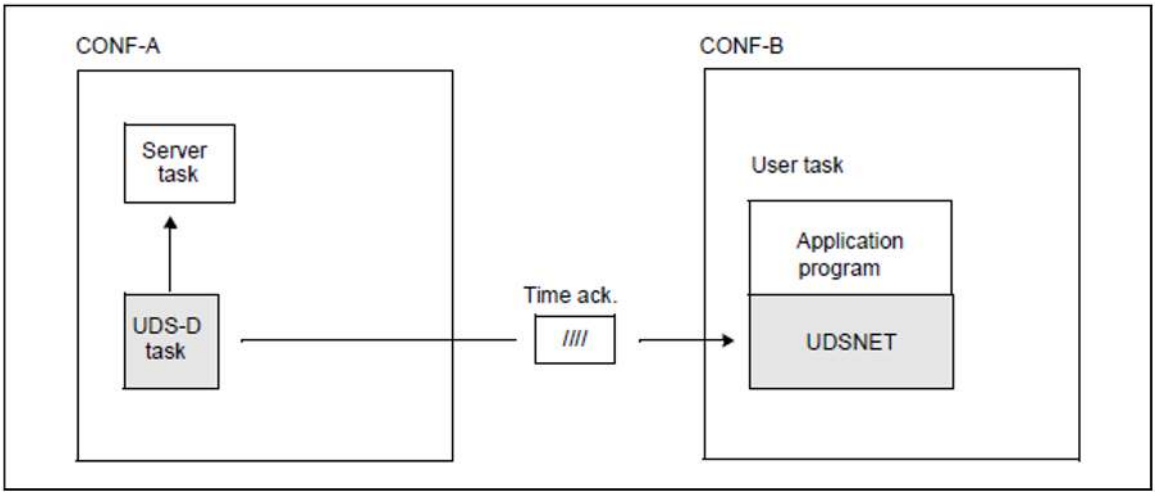


Figure 25: Monitoring the logical connection to the remote application program

UDSNET waits for a message from the remote UDS-D task for a fixed time period, which depends on the value of PP CHCKTIME.

If this time period expires without a message being received from the remote UDS-D task and without any message indicating a lost logical connection, UDSNET clears the connection to the remote UDS-D task, and then reconnects and repeats the last DML statement.

If UDSNET does not receive a response again, it rolls back the transaction, and status code 122, “Transaction prematurely canceled”, is sent to the application program.

14.5.4.2 Monitoring secondary subtransactions that are not in the PTC state

If a secondary subtransaction that is not in the PTC state has not received any further message form its primary subtransaction within a time period determined by PP CHCKTIME, the UDS-D task queries the corresponding UDS-D task of the primary subtransaction‘s configuration to check whether the primary subtransaction involved still exists. If it does not, the secondary subtransaction is rolled back. If the UDS-D task of the primary subtransaction is not accessible, the secondary subtransaction is retained.

14.5.4.3 Monitoring secondary subtransactions in the PTC state

Secondary subtransactions in the PTC state wait for the FINISH or FINISH WITH CANCEL statement from the associated primary subtransaction.
If a secondary subtransaction loses the logical connection to its primary subtransaction in this phase, the automatic PTC monitoring performed by UDS-D becomes critically important.

A secondary subtransaction in the PTC state waits for a message from the primary subtransaction for a defined time interval, which depends on PP CHCKTIME. Following this interval, the UDS-D task of the secondary subtransaction automatically tries to determine the state of the associated primary subtransaction. If this is successful, the secondary subtransaction is handled in the same way as the primary subtransaction. If the attempt fails, the secondary subtransaction is handled as determined by the value of the DBH load parameter PTCSYNCH for the current session, i.e. if PTCSYNCH is set to ABORT or COMMIT, the secondary subtransaction in the PTC state is terminated with ABORT or COMMIT, and if PTCSYNCH is set to WAIT, it remains in the PTC state.

If desired, the DAL command &SYNCHRONIZE DISTRIBUTION can be used by the database administrator to speed up the process of checking the configuration of the primary subtransaction. If the configuration of the primary subtransaction cannot be reached even after &SYNCHRONIZE DISTRIBUTION, all secondary subtransactions for which the state of the primary subtransaction cannot be determined are handled as set in the PTCSYNCH value for the current session on completion of the &SYNCHRONIZE DISTRIBUTION command.

> ¡ If the DBH load parameter PP PTCSYNCH is not set to (WAIT,WAIT), the cross-configuration consistency or the consistency between UDS/SQL and openUTM can no longer be guaranteed (see also section “Terminating the PTC state”).

14.5.4.4 Monitoring primary subtransactions that were terminated by the database administrator

If the database administrator terminates primary subtransactions “manually” by using the DAL command COMMIT or ABORT,OPTION=PTC, the associated secondary subtransactions are automatically notified via the timer and terminated in the same way as the primary subtransaction.

14.5.5 Starting and terminating UDS-D mode

The following sections describe how you can start, terminate, and subsequently restart UDS-D mode.

14.5.5.1 Starting UDS-D mode

When the DBH is started with the independent DBH, the DBH load parameter `PP DISTRIBUTION` can be used to control whether or not UDS-D mode is to be started for a configuration (see section “The session”). If UDS-D mode is started, the configuration may participate in it. Participating in the UDS-D mode means that local application programs can start distributed transactions and can process DML statements of remote application programs.

- `PP DISTRIBUTION=NO` (default): This means that the configuration does not participate in UDS-D mode in the initiated session. In other words, local application programs cannot start distributed transactions and cannot process secondary subtransactions from remote application programs. Any attempt to access a database of this configuration from a remote application program is rejected with status code 141.
- `PP DISTRIBUTION=START`: The configuration will automatically participate in UDS-D mode, as soon as the DBH is started.
- `PP DISTRIBUTION=STANDBY`: The configuration is prepared for UDS-D mode, but does not participate in it until the DAL command `&START DISTRIBUTION` is issued by the database administrator to start the UDS-D mode.

If you have set the DBH load parameter `PP DISTRIBUTION` to START or STANDBY, the master task will, on starting up, place the ENTER file for the UDS-D task UDSCT under the user ID in which you started the DBH under the following name:

```
UDS.ENTER.tsn.CT000
```

tsn Task sequence number of the master task

If `PP DISTRIBUTION=START` is specified, the ENTER job is started immediately.
The ENTER file continues to exist until DBH termination, even if the UDS-D mode is terminated beforehand.

Further DBH load parameter to control UDS-D mode can be found in section “DBH load parameters for UDS-D” and in chapter “DBH load parameters”.

> If PP LOG=NO, applies, the DBH will not allow local application program to start distributed UPDATE transactions, since no logging and no session restart is possible.
> Distributed RETRIEVAL transactions, by contrast, are allowed with PP LOG=NO.

14.5.5.2 Terminating UDS-D mode

UDS-D mode can be terminated by

- terminating the DBH or
- terminating only UDS-D mode.

Terminating the DBH

The DBH can be terminated with one of the following DAL commands:

- CLOSE RUN-UNITS (normal DBH termination)
- CLOSE CALLS (accelerated, but normal DBH termination using FINISH WITH CANCEL for all open transactions).
  UDS/SQL can terminate the DBH with CLOSE RUN-UNITS or CLOSE CALLS only if none of the subtransactions are in the PTC state.
  The database administrator can, however, force the DBH to terminate even if there are subtransactions in the PTC state (see section “Terminating the PTC state”).
- %TERM (instant DBH abort, without terminating open transactions)
  Subtransactions in the PTC state remain in the same state and are subsequently handled in a warm start as defined by the value of the DBH load parameter PP PTCSYNCH (see “Effects of the PTC state during a warm start”).

The DAL commands to terminate the DBH are described in section “Terminating the DBH”.

Terminating UDS-D mode

UDS-D mode can be terminated with DAL command &CLOSE DISTRIBUTION. When all subtransactions have been closed, UDS/SQL terminates the UDS-D task UDSCT. Subtransactions in the PTC state that could not be terminated within a time interval defined by PP CHCKTIME remain in the PTC state. Once the UDS-D task has been terminated, UDS/SQL will accept neither the remote processing chains of a local application program, nor the secondary subtransactions of a remote application program.

If desired, the terminated UDS-D mode can also be restarted for the same session (see section “Restarting UDS-D mode”).

> ¡ Local application programs that are part of a configuration in which the UDS-D task has terminated abnormally can usually continue to access remote configurations.

14.5.5.3 Restarting UDS-D mode

UDS-D mode can usually be restarted within a session after a normal or abnormal termination by using the DAL command &START DISTRIBUTION. This causes the existing UDS-D tables to be reused.
Secondary subtransactions that are still open from the previous UDS-D mode are handled as follows:

- Secondary subtransactions that are not in the PTC state are rolled back.
- Secondary subtransactions in the PTC state are terminated in the same way as the associated primary subtransaction. If the state of the primary subtransaction cannot be determined, the secondary subtransaction is handled as defined by the value of the DBH load parameter PP PTCSYNCH for the current session.
  - If PP PTCSYNCH=(...,ABORT) or =(...,COMMIT), the secondary subtransaction is terminated with ABORT or COMMIT.
  - If PP PTCSYNCH=(...,WAIT), the secondary subtransaction remains in the PTC state.

14.5.5.4 Examples for UDS-D mode

The following runtime example illustrates how you can start, terminate, and then restart UDS-D mode.

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONFSHPG
/CREATE-FILE FILE-NAME=CONFSHPG.DBSTAT
/CREATE-FILE FILE-NAME=CONFSHPG.DBSTAT.SAVE
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.9C00
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D, VERSION=02.9C00
/START-UDS-DBH
```

```
...
%  BLS0523 ELEMENT 'UDSSQL', VERSION '02.9C00', TYPE 'L' FROM LIBRARY
':GIN1:$TSOS.SYSLNK.UDS-SQL.029' IN PROCESS
%  BLS0524 LLM 'UDSSQL', VERSION '02.9C00' OF '2019-01-29 09:31:40' LOADED
PP ADM=REMOTE
PP DBNAME=SHIPPING
PP LOG=PUBLIC
PP MAXDB=4
PP PARLIST=YES
PP PRIVACY-CHECK=OFF
PP DISDB=3
PP DISTABLE=UDSDBB.VT.CONFSHPG
PP DISTRIBUTION=START
PP END
%  %  UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS (OPI6746,09:31:41/4TUD)
%  4TUD: UDS-PUBSET-JV:  :IUDS:$XXXXXXXX.PUBSDECL.PUBS
%  4TUD: PUBSETS:       ABN2
%  4TUD: DEFAULT PUBSET: IUDS
%  4TUD: ----------------------------------------
%  UDS0347 UDS ADMINISTRATION: APP.NAME = CONFSHPG (OPDI239,09:31:41/4TUD)
%  UDS0722 UDS ORDER ADD RLOG 150628094330 IN EXECUTION (OPCC283,09:31:41/4TUD)
4TUD: MAXDB         =      4
4TUD: TRANSACTION   = (    4,   1)
4TUD: SUBSCHEMA     =      1
4TUD: SERVERTASK    =      1
4TUD: 2KB-BUFFER-SIZE=     1
4TUD: 4KB-BUFFER-SIZE=     1
4TUD: 8KB-BUFFER-SIZE=     0
4TUD: CP-SIZE       =   1024
4TUD: CUP-SIZE      =   1024
4TUD: SIP-SIZE      =   1024
4TUD: DIP-SIZE      =   1024
4TUD: DISDB         =      3
4TUD: CHCKTIME      =     60
4TUD: DEADTIME      =     60
4TUD: DISTRIBUTION  =  START
4TUD: PTCSYNCH      =  (WAIT  ,WAIT  )
4TUD: LOG           = PUBLIC
4TUD: LOG-2         = NO
4TUD: LOG-SIZE      = (   192,   192)
4TUD: RESERVE       = NONE
4TUD: WARMSTART     = STD
4TUD: DBDCSYNCH     = YES
4TUD: DEACT         = YES
4TUD: STDCKPT       = NO
4TUD: ADM           = REMOTE
```

```
4TUD: CPU           = MONO-PROCESSOR
4TUD: DISTABLE      = UDSDBB.VT.CONFSHPG
4TUD: SQL           =     4
4TUD: SQL-LIMIT     =    10
4TUD: TASKLIB       = NO
4TUD: DUMP          = ALL
4TUD: MPSEG         = STD
4TUD: UCON          = C'<U  ',MSG
4TUD: SECLEVEL      = NO      ,NO-AUDIT
4TUD: LOCK          = STD
4TUD: TA-ACCESS     = STD
4TUD: WAIT          = EVENT
4TUD: RESULT-DELAY  =     0
4TUD: SCHEDULING    = SYMMETRIC
4TUD: IO            = ASYNC
4TUD: ORDER-DBSTATUS= STD
4TUD: PRIVACY-CHECK = OFF
4TUD: BCAM-PREFIX   = SUD$
4TUD: CONFNAME      = $XXXXXXXX.CONFSHPG
...
```

/EXECUTE-DAL-CMD CMD=&CLOSE DISTRIBUTION

```
%  UDS0220 UDS RECEIVED COMMAND: &CLOSE DISTRIBUTION (OPOX073,09:33:21/4TUD)
%  UDS0832 UDS-D: COMMAND EXECUTED   ((CTCC012,09:33:21/4TUF)
%  UDS0807 UDS-D: TERMINATION INITIATED   (CTCC351,09:33:21/4TUF)
%  UDS0808 UDS-D TERMINATING  (OPCC352,09:33:21/4TUD)
%  UDS0809 UDS-D NORMAL TERMINATION (2019-01-29) (CTCC353,09:33:21/4TUD)
...
```

/EXECUTE-DAL-CMD CMD=&SAVE DISTRIBUTION,FILE=VT.EXAMPLE.SAVE

```
%  UDS0220 UDS RECEIVED COMMAND: &SAVE DISTRIBUTION,FILE=VT.EXAMPLE.SAVE (OPOX073,09:33:21
/4TUD)
%  UDS0808 UDS-D TERMINATING  (OPCC352,09:33:21/4TUD)
...
```

/EXECUTE-DAL-CMD CMD=&START DISTRIBUTION

```
%  UDS0220 UDS RECEIVED COMMAND: &START DISTRIBUTION (OPOX073,09:33:21/4TUD)
%  UDS0832 UDS-D: COMMAND EXECUTED  (OPCC382,09:34:21/4TUD)
%  UDS0801 UDS-D STARTING V2.9/2019-01-29 (CTCC345,09:33:21/4TUD)
%  UDS0823 UDS-D: INPUT FILE FOR DISTRIBUTION TABLE UDSDBB.VT.CONFSHPG READ (CTCC386,09:33:21
/4TUD)
%  UDS0805 UDS-D READY   (CTCC349,09:33:21/4TUD)
...
```

//EXECUTE-DAL-CMD CMD=&SAVE DISTRIBUTION,FILE=VT.EXAMPLE.SAVE

```
%  UDS0220 UDS RECEIVED COMMAND: &SAVE DISTRIBUTION,FILE=VT.EXAMPLE.SAVE (OPOX073,09:33:21
/4TUD)
%  UDS0832 UDS-D: COMMAND EXECUTED   (OPCC352,09:33:21/4TUD)
...
```

Input file for the distribution table VT.EXAMPLE.START

```
DB  DB11   CONF3  PROZ1
DB  DB12   CONF3  PROZ1
*
* DATABASE RECORDS CONF-4
DB  DB21   CONF4  PROZ2
DB  DB22   CONF4  PROZ2
*
* SUBSCHEMA RECORDS CONF-3
SS  SS111  DB11
SS -SS112  DB11
*
SS  SS121  DB12
SS  SS122  DB12
*
* SUBSCHEMA RECORDS CONF-4
SS  SS21   DB21
SS  SS22   DB22
```

The following runtime example shows how multiple DB configurations are accessed.

The program DMLTEST accesses the local configuration CONFVERS (subschema ADMIN) and the remote configuration CONFCSTM (subschema MANAGEMENT).

/CALL-PROCEDURE P.DMLTEST.IND

/BEGIN-PROC LOG=*ALL,PAR=*YES(PROC-PAR=(&CONFNAME))
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=&CONFNAME
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=LMS.SSITAB

```
&&CONFNAME=CONFVERS
```

/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CONFSHPG
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.9C00
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D, VERSION=02.9C00

/START-UDS-DMLTEST

```
...
DBH IND
LANG COB
DISPLAY RCODE,COND=RCODE NE C'00000'
PROT ON
DISPLAY RECA,L=80
SYS
```

SUBSCHEMA ADMIN
READY USAGE-MODE UPDATE;E

```
RECORD - AREA     :
.....................................................................
```

FETCH LAST COLORS;E

```
RECORD - AREA     :
25LILA              ..........................................................
```

M RECA,26MAGENTA
STORE COLORS;E

```
RECORD - AREA     :
26MAGENTA           ..........................................................
```

SUBSCHEMA MANAGEMENT
READY USAGE-MODE UPDATE;E

```
RECORD - AREA     :
26MAGENTA           ..........................................................
```

FETCH LAST CUSTOMER;E

```
RECORD - AREA     :
YMEIER              ..........................................................
```

M RECA,SMITTY
MODIFY K-NAME;E

```
RECORD - AREA     :
SCHMIDTCHEN .....................................................................
```

FETCH LAST CUSTOMER;E

```
RECORD - AREA     :
SCHMIDTCHEN.....................................................................
```

SUBSCHEMA ADMIN
FETCH LAST COLORS;E

```
RECORD - AREA     :
26MAGENTA           ..........................................................
```

FINISH;E

```
RECORD - AREA     :
26MAGENTA           ..........................................................
```

STOP

```
DMLTEST NORMAL TERMINATION
```

Input file for the distribution tables of both configurations

```
DB  SHIPPING    CONFSHPG  XXXXXXXX
DB  CUSTOMER    CONFCSTM  XXXXXXXX
DB  ARTICLES    CONFART   YYYYYYYY
*
SS  ADMIN       SHIPPING
SS  MANAGEMENT  CUSTOMER
SS  SUBART      ARTICLES
```

Configuration of the primary subtransaction

//EXECUTE-DAL-CMD CMD=DISPLAY USERS

```
%  UDS0347 UDS ADMINISTRATION: LOGON    = UAD@4TE7 (OPDM239,09:33:13/4TUC)
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY USERS (OPOX073,09:33:13/4TUC)
4TUC: PROGRAM   TSN/TERM     RUNUNIT-ID  STATE    PTC  FC   MR-NR DLY
4TUC: ------------------------------------------------------------
4TUC: DMLTEST   TSN 4TUJ           1    USER     -   134    1    -
4TUC:
4TUC: UDS/SQL V2.9     1 USER  OF CONFCSTM
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND   (OPCC074,09:33:13/4TUC)
//EXECUTE-DAL-CMD CMD=DISPLAY 1
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY 1 (OPOX073,09:33:13/4TUC)
4TUC: PROGRAM-NAME:       DMLTEST
4TUC: TSN:                4TUJ
4TUC: RUNUNIT-ID / -STATE:        1 / USER
4TUC: MAINREF-NR / -STATE:        1 / USED
4TUC: FC:                 134
4TUC: PTT-PROC-NAME:      IBAPROD1
4TUC: PTT-CONF-NAME:      CONFSHPG
4TUC: PTT-RLOG-DATE:
4TUC: PTT-RUNUNIT-ID:            1
4TUC: LOCAL-TT-STATE:     STT
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:33:13/4TUC)
```

Configuration of the secondary subtransaction

//EXECUTE-DAL-CMD CMD=DISPLAY USERS

```
%  UDS0347 UDS ADMINISTRATION: LOGON    = UAD@4TE7 (OPDM239,09:33:17/4TUD)
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY USERS (OPOX073,09:33:17/4TUD)
4TUD: PROGRAM   TSN/TERM    RUNUNIT-ID  STATE    PTC  FC   MR-NR DLY
4TUD: ------------------------------------------------------------
4TUD: DMLTEST   TSN 4TUJ           1    USER     -   134    1    -
4TUD:
4TUD: UDS/SQL V2.9     1 USER  OF CONFSHPG
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:33:17/4TUD)
```

//EXECUTE-DAL-CMD CMD=DISPLAY 1

```
%  UDS0220 UDS RECEIVED COMMAND: DISPLAY 1 (OPOX073,09:33:17/4TUD)
4TUD: PROGRAM-NAME:       DMLTEST
4TUD: TSN:                4TUJ
4TUD: RUNUNIT-ID / -STATE:        1 / USER
4TUD: MAINREF-NR / -STATE:        1 / USED
4TUD: FC:                 134
4TUD: PTT-PROC-NAME:      IBAPROD1
4TUD: PTT-CONF-NAME:      CONFSHPG
4TUD: PTT-RLOG-DATE:      190129083140
4TUD: PTT-RUNUNIT-ID:             1
4TUD: LOCAL-TT-STATE:      PTT
%  UDS0218 UDS COMPLETED EXECUTION OF DAL COMMAND  (OPCC074,09:33:17/4TUD)
```

14.5.6 Effects of the PTC state

The following information is saved in the RLOG file of each respective subtransaction during the two-phase commit:

- “Transaction in PTC state”: if a subtransaction of the transaction involved is in the PTC state
- “Transaction terminated with FINISH” or “Transaction terminated with FINISH WITH CANCEL“: if the primary subtransaction was terminated with FINISH or FINISH WITH CANCEL.

See section “Ensuring cross-configuration consistency with a two-phase commit”.

Since the RLOG file is maintained on a session-specific basis, it may not be possible to check the configuration of the primary subtransaction after a normal DBH termination to determine how the primary transaction was terminated. Consequently, information on how the transaction terminates is also saved in the DB status file, which is maintained on a configuration-specific basis.

The PTC state is a critical phase of the two-phase commit. The DBH load parameter PP PTCSYNCH can be used to control how a subtransaction in the PTC state is handled.

> ¡ The DBH load parameter PP PTCSYNCH should always be set to (WAIT,WAIT), since this is the only way to guarantee cross-configuration consistency or consistency between UDS/SQL and openUTM! The database administrator should therefore avoid changing the PTCSYNCH value to ABORT or COMMIT unless he or she can make an informed decision about the cross-configuration consistency (see also section “Terminating the PTC state”).

The PTC state has a significant effect:

- during the session
- during a warm start
- on terminating the DBH
- on the application program when the logical connection is lost

Effects of the PTC state during a session

During a session, the PTC state has a significant effect when:

- restarting UDS-D mode after a normal or abnormal termination.
- altering the database configuration (see section “Altering the DB configuration: notes for UDS-D”).
  If there are subtransactions in the PTC state, the databases involved cannot be detached consistently from the DB configuration.
- writing a checkpoint (see “Writing checkpoints (CHECKPOINT)”).
  When checkpoints are to be written, it is important to ensure that the connection to the configuration of the primary subtransaction or to openUTM exists. This is because UDS/SQL will only write checkpoints for a database if all transactions that have updated this database have terminated.

If attempts to write checkpoints or to detach or attach databases have already been initiated, but cannot be completed because of subtransactions in the PTC state, the database administrator can:

- use &SYNCHRONIZE DISTRIBUTION to have the secondary subtransactions check how the primary subtransaction was terminated. If UDS/SQL cannot determine how this occurred, the secondary subtransaction remains in the PTC state.

- use DISPLAY USERS to check the transaction IDs of the subtransactions in the PTC state and then terminate these transactions with the following DAL commands:

  `ABORT` *transaction-id*`,OPTION=PTC`
  or
  `COMMIT` *transaction-id*

  > If the database administrator intervenes with ABORT,OPTION=PTC or COMMIT, cross-configuration consistency or the consistency between UDS/SQL and openUTM cannot be guaranteed (see section “Terminating the PTC state”).

- use %TERM

  to abort the session and then restart without the affected database.

Effects of the PTC state during a warm start

In the case of a warm start, all open transactions are rolled back. Subtransactions in the PTC state, if any, are handled in accordance with the DBH load parameter PP PTCSYNCH for a warm start.

If the value of this parameter has been set to WAIT for a primary subtransaction, the database involved is not attached during a warm start until openUTM requests the termination of the transaction.

If the PP PTCSYNCH parameter is set to WAIT for a secondary subtransaction, UDS/SQL will try to reach the associated primary subtransaction. The following conditions must be satisfied for this purpose:

- UDS-D mode must have been started in the respective configurations of the primary and secondary subtransactions.
- The distribution tables must reflect the conditions on the network.
- There must be one configuration with the configuration name under which the primary subtransaction was running before the session abort.
  This configuration may have a different composition or be located on another host, provided the distribution tables have been adjusted accordingly.
- The DB status file must be present in the respective configurations of the primary and secondary subtransactions.

If the state of the associated primary subtransaction cannot be determined, the secondary subtransaction remains in the PTC state, and the database in question is not attached.

Figure 26 shows the processing sequence of a warm start when there are secondary subtransactions in the PTC state.

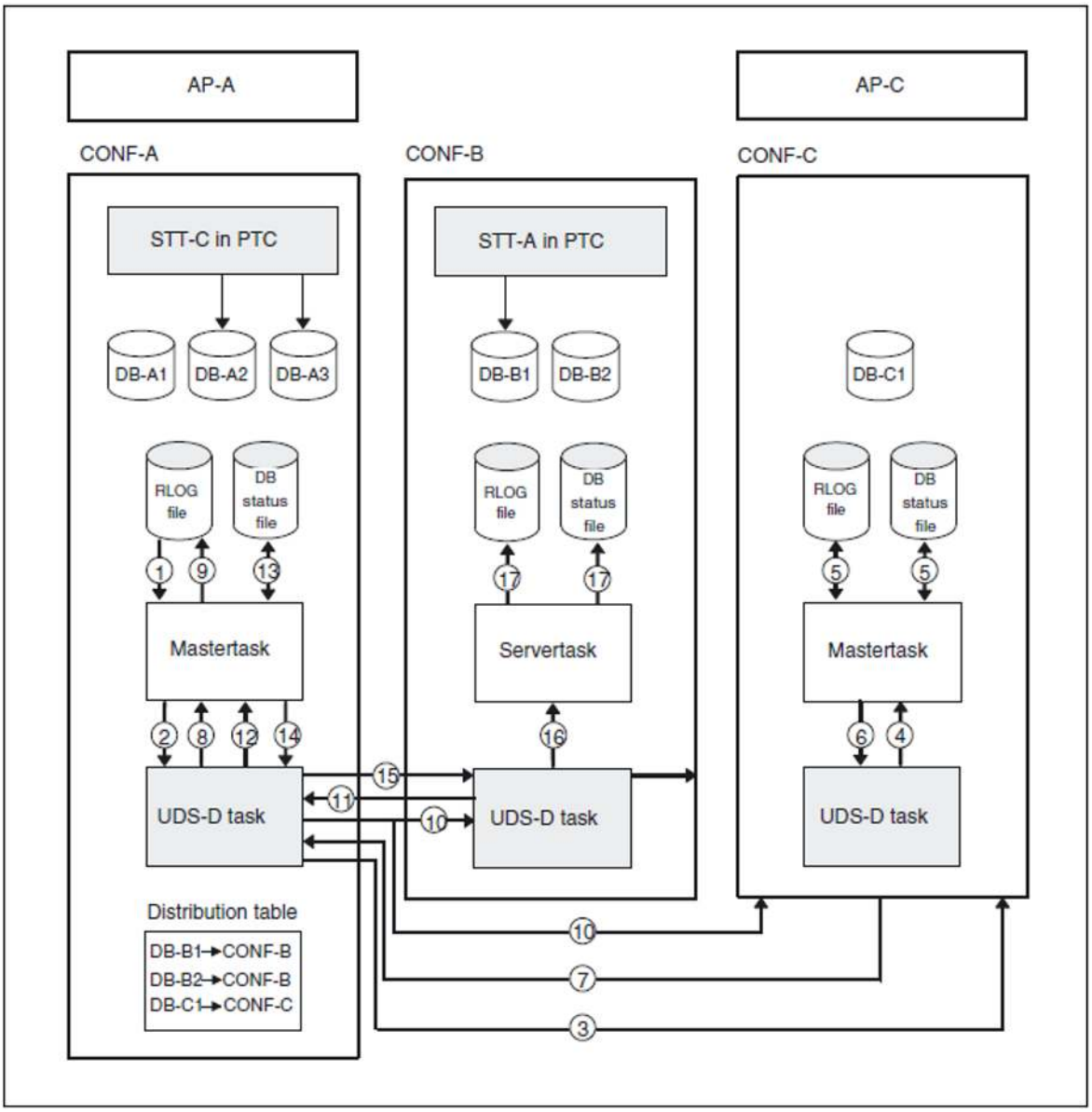


Figure 26: Warm start with secondary subtransactions in the PTC state

Following a session abort, the databases DB-A1, DB-A2 and DB-A3 are to be started in the configuration CONF-A. The databases DB-A2 and DB-A3 require a warm start, since the secondary subtransaction STT-C, which was initiated in the application program AP-C of the configuration CONF-C, is still in the PTC state. The RLOG file and DB status file for CONF-C contains an entry to indicate whether the primary subtransaction PTT-C was terminated with FINISH or FINISH WITH CANCEL.

1. During the warm start of configuration CONF-A, a transaction in the PTC state is found in the RLOG file.
2. The master task passes the warm start information stored in the RLOG file to the UDS-D task.
3. The UDS-D task uses the warm start information to determine the host processor and the configuration of the primary subtransaction associated with STT-C. It then sends a message to the remote UDS-D task with a request to determine how the primary transaction was terminated.
4. This message is passed by the remote UDS-D task to its master task.
5. The master task checks the DB status file to determine how the primary transaction was terminated. If this file is not current, it looks in the RLOG file.

6. and 7.

   The result is passed to the UDS-D, and then forwarded on to the UDS-D task of the secondary subtransaction.

8. The UDS-D task passes the result to the master task.

9. The secondary subtransaction STT-C is terminated in the same way as the primary subtransaction.

10. The UDS-D task of CONF-A contacts all configurations listed in its distribution table to notify them that the configuration CONF-A may be addressed again.

11. Since the secondary subtransaction STT-A in the configuration CONF-B is still in the PTC state, the UDS-D task of CONF-B sends a message to the UDS-D task of the primary subtransaction to determine the state of the primary subtransaction.

12. and 13.

    See 4. and 5.

14. and 15.

    See 6. and 7.

16. Depending on the result, the UDS-D task sends the FINISH or FINISH WITH CANCEL request to the server task.

17. The termination of the transaction is logged in the RLOG file.

If a secondary subtransaction in the PTC state cannot be terminated in the same way as its primary subtransaction in a warm start, the secondary subtransaction remains in the PTC state, and the database in question is not attached during the warm start. The database administrator is informed of the other subtransactions involved by means of a message.

If there are databases that cannot be attached during a warm start because of a subtransaction in the PTC state, you will need to wait until the configuration of the primary subtransaction can be accessed again or until openUTM requests the termination of the transaction. Alternatively, you may also use the DAL command MODIFY to set the PTCSYNCH value for a warm start to ABORT or COMMIT.

> ¡ If the DBH load parameter PP PTCSYNCH is not set to (WAIT,WAIT), the cross-configuration consistency or the consistency between UDS/SQL and openUTM cannot be guaranteed (see section “Terminating the PTC state”).

Effects of the PTC state on terminating the DBH

- DBH termination with CLOSE RUN-UNITS or CLOSE CALLS

  UDS-D waits until all secondary subtransactions in the PTC state are terminated automatically or until openUTM requests the end of the transaction.
  If the transactions to not terminate automatically, the database administrator can:

- use &SYNCHRONIZE DISTRIBUTION to have the secondary subtransactions check how the primary subtransaction was terminated. If UDS/SQL cannot determine how this occurred, the secondary subtransaction remains in the PTC state.
- use DISPLAY USERS to check the transaction IDs of the subtransactions in the PTC state and then terminate these transactions with the following DAL commands:

  `ABORT` *transaction-id*`,OPTION=PTC`
  or
  `COMMIT` *transaction-id*

  If no further subtransactions are open, the DBH is terminated.

  > If the database administrator terminates the subtransactions with ABORT,OPTION=PTC or COMMIT, cross-configuration consistency or the consistency between UDS/SQL and openUTM cannot be guaranteed (see section “Terminating the PTC state”).

- use %TERM to abort the session

  Subtransactions in the PTC state remain in the PTC state. The special aspects to be noted for a warm start in this case can be found under “Effects of the PTC state during a warm start”.

  > Following a session abort, databases that were rendered inconsistent during the session should be attached with a warm start as soon as possible to ensure proper communication between the DB configurations.

Effects of the PTC state on an application program when a logical connection is lost

If the logical connection to a remote configuration is lost during the termination phase of a distributed transaction and cannot be established again, the effects on the application program will depend on which phase of the two-phase commit the primary subtransaction is in.

- The user task has sent the PETA statement to the secondary subtransactions, but the primary subtransaction itself has not yet terminated.
  The application program receives status code 122, “Transaction prematurely canceled”, from the connection module.
- The application program has initiated a FINISH that was executed successfully.
  In this case, the application program receives status code 000: “DML statement executed successfully“, from the connection module.

  In the configuration of primary subtransaction, the transaction is treated as terminated, regardless of whether or not a secondary subtransaction in the PTC state still exists in a remote configuration.
- If the FINISH of the primary subtransaction executes with errors, the application program receives the status code of the failed FINISH.

14.5.7 Administering multiple configurations

There are two ways of administering the DB configurations involved in a distribution or their database handlers:

- local administration
- central administration

It is also possible to administer some DB configurations locally and some centrally.

14.5.7.1 Local administration

Every DB configuration is administered via a separate console and via one of the following interfaces:

- via the UDSADM administration program (see section “Administration of UDS/SQL via UDSADM”).
- via the BS2000 command `INFORM-PROGRAM MSG='dal-command' ["comments"]` (see section “Administration of UDS/SQL via INFORM-PROGRAM”).
- via the DCAM interface for administration (see section “Administration of UDS/SQL via DCAM”).

The database administrator language DAL is described in section “The Database Administrator Language DAL”. An overview of the DAL commands for distributed databases can be found in section “The Database Administrator Language DAL for UDS-D”.

14.5.7.2 Central administration

In the case of central administration, **all** DB configurations are addressed from **a single** console, i.e. every DBH involved can be started, administered and terminated from the same console.

It is also possible to administer multiple DB configurations centrally

- with UDSADM via OMNIS
- by using the software product OMNIS as of OMNIS Version 4.0.
  A detailed description of OMNIS can be found in the OMNIS manuals “Functions and Commands” and “ Administration and Programming”.
- via an administration program that uses the DCAM interface
  (see section “Administration of UDS/SQL via DCAM”).

Central administration using OMNIS

OMNIS communicates messages between the database administrator and the DBHs to be managed, regardless of which hosts are running these DBHs.

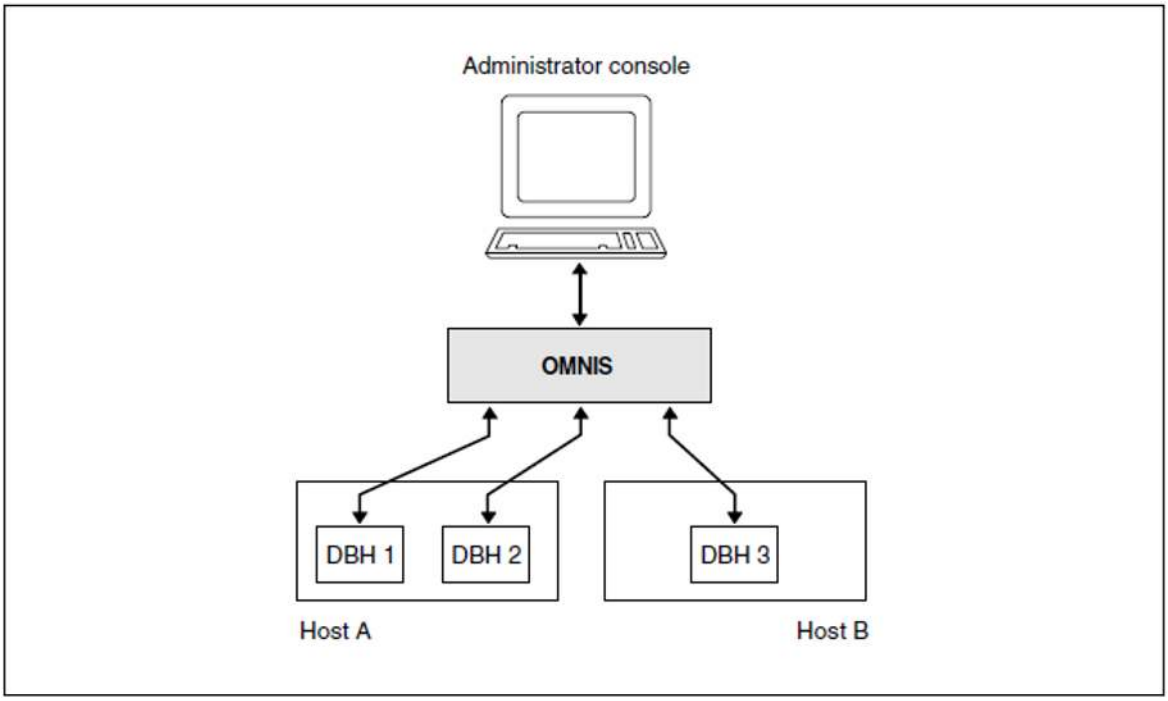


Figure 27: Central administration using OMNIS

Starting and administering DBHs with OMNIS

For operational security reasons, it is advisable to start UDS/SQL as an ENTER job and to perform administration tasks via UDSADM or the DCAM interface.

All DBHs must be started with the DBH load parameter PP ADM=REMOTE (see PP ADM in chapter "DBH load parameters"). This causes every DBH to open a DCAM application for administration with the application name *confname*.
The database administrator connects OMNIS with these DCAM applications or starts UDSADM in multiple tasks and can then address all DBHs via OMNIS from one console.

If the connection to OMNIS is down, the database administrator can also manage the DB configuration locally. Central administration can be resumed as soon as OMNIS is available again, assuming that the configuration is not being managed locally.

> ℹ In the OMNIS administration concept, all data terminals have the same privileges. Consequently, it is only possible to provide limited protection against unauthorized interference with the UDS/SQL administration.

14.5.8 Altering the DB configuration: notes for UDS-D

The section “Altering the DB configuration during the session” can be found on "Defining and altering the DB configuration".

The following must be noted for UDS-D:

- If you have modified any DB configurations, you may have to adapt the distribution tables accordingly.
- If there are secondary subtransactions in the PTC state, the databases involved cannot be detached from the DB configuration in a consistent state (see section “Effects of the PTC state”).
  You should therefore avoid exclusions after a session abort or perform a warm start on the inconsistent databases as soon as possible to ensure proper communication between the DB configurations.

In the case of an emergency, the PTC state can also be terminated “manually” (see section “Terminating the PTC state”).

14.5.9 Using the UDS/SQL configuration on a virtual host

A UDS/SQL configuration can also be used on a virtual host. This enables the distribution rules to be defined independently of real host names. To permit fault-free UDS-D operation, the communications partners (DBH and user tasks) must be assigned to the virtual host. If more than one UDS/SQL configuration exists on a real host, these can execute on different virtual hosts in UDS-D operation.
It must be pointed out that you should assign a unique BCAM prefix for each UDS/SQL configuration using the DBH start parameter BCAM-PREFIX (see also section “Defining a prefix for user tasks in UDS-D distributions (BCAM-PREFIX)” in chapter "DBH load parameters"). The UDS/SQL configuration and application are assigned to a special virtual host using an entry in the BCAM configuration file for applications (the default file name is SYSDAT.BCAM.APPLICATIONS).

If the application is operated in openUTM transaction mode, the DCAM name with which the application can be addressed in transaction mode should also be assigned to the same virtual host.

Names of user tasks

DCAM applications whose names consist of a prefix and the TSN are created for UDS/SQL communication which is distributed with UDS-D between the user task and the remote configuration.
By default the name is SUD$*tsn*. This name complies with the naming convention in BS2000 (S=SYSTEM, UD=product abbreviation for UDS/SQL).

However, you can also specify the prefix of the name in the load parameter BCAM-PREFIX of the independent DBH at the start of a session. This enables the user tasks of different configurations to be distinguished on the basis of the BCAM prefix and also to be assigned to different virtual hosts (see also section “Defining a prefix for user tasks in UDS-D distributions (BCAM-PREFIX)” in chapter "DBH load parameters").

> ¡ If a value which differs from the default value is specified for the load parameter BCAM-PREFIX, it is recommended that you use at least UDS-D V2.6 on all remote configurations which participate via UDS-D, also in those participating configurations which do not execute on virtual hosts, otherwise on configurations with older UDS-D versions bottlenecks can occur which can also hinder other DML processing.

Assigning the communications partners

The communications partners are assigned to the virtual host in the BCAM configuration file for applications (the default file name is SYSDAT.BCAM.APPLICATIONS). For details, please refer to the “BCAM” manual.

Example:

The local UDS/SQL configuration UKONF1, together with the applications assigned to it, is to be connected to the virtual host VIRT01. The applications in transaction mode are all addressed with the names APPLIK11 through APPLIK99.
In this constellation the DCAM applications which take over distributed communication between the application and the remote configuration should be assigned the BCAM prefix VKO1.

Name of the UDS/SQL configuration: UKONF1
Name of the virtual host: VIRT01
BCAM prefix for communication between application and DBH: VKO1
Names of the applications for transaction mode: APPLIK11, ..., APPLIK99

The following entries must then be made in the BCAM configuration file:

```
NEA UKONF1* VIRT01
NEA APPLIK* VIRT01
NEA VKO1* VIRT01
```

However, if a special BCAM prefix for UDS-D communication between an application and a remote UDS/SQL configuration is to be dispensed with, the following entry (specification of the default prefix) must be contained in the BCAM communication file instead of the last entry above:

```
NEA SUD$* VIRT01
```

14.6 DBH load parameters for UDS-D

The following DBH load parameters, which are important for UDS-D, are processed bythe independent DBH:

Parameter	Default value	Meaning
PP BCAM-PREFIX=*prefix*	SUD$	Define prefix for names of user tasks which execute on virtual hosts
PP CHCKTIME=*n*	60	Specify time interval in seconds for connection and transaction monitoring of the UDS-D task *n*=60..900
PP DEADTIME=*n*	60	Time interval in seconds for resolution of cross-configuration deadlocks or deadlocks in connection with openUTM *n*=5..900
PP DISDB=*n*	1	Maximum number of remote databases that can be addressed per transaction; *n*=1..32
PP DISTABLE=[:*catid*:][$*userid*.] *filename*	-	Define input file for creating the distribution table
PP DISTRIBUTION={NO | STANDBY | START}	NO	Control participation in UDS-D mode: NO: UDS-D mode is not possible STANDBY: UDS-D mode is ready and can be subsequently started with &START DISTRIBUTION START: UDS-D mode is started
PP PTCSYNCH= ([{WAIT | ABORT | COMMIT}] [,[{WAIT | ABORT | COMMIT}]])	(WAIT,WAIT)	Control handling of transactions in the PTC state; the first value applies to a warm start; the second applies during a running session if the state of the primary subtransaction cannot be determined

PP TRANSACTION={*n* | ([*n*][,*m*])}	(4,1)	*n*: Maximum number of simultaneously active transactions and user tasks *n* = 1..225; *m*: Maximum number of secondary subtransactions that can be processed concurrently by this DBH *m* = 1..*n* and *m* <= *n*

Table 33: DBH load parameters for UDS-D

A detailed description of the DBH load parameters can be found in chapter 3 on "DBH load parameters".

14.7 The Database Administrator Language DAL for UDS-D

The following DAL commands, which are important for UDS-D, are processed by the independent DBH.

All DAL commands that are only known to UDS-D begin with the special character “&”. These DAL commands (except for &START DISTRIBUTION) are rejected if UDS-D mode has not been started. Detailed descriptions of the “&” DAL commands can be found in the section describing the other UDS-D DAL commands in alphabetical order.

DAL command	Meaning
ABORT {*transaction-id*[,OPTION=PTC] | ALL}	Roll back the specified open transactions If OPTION=PTC is specified: without taking cross-configuration consistency or UDS/SQL and openUTM consistency into account
&ADD DISTRIBUTION, {NODE=*processor-name*, CONF=*confname*[,DB=*dbname*] | DB=*dbname*,SS=*subschema-name* | FILE=*file-name* }	Add new entries to the distribution table
&CHANGE DISTRIBUTION, NODE=*processor-name*, CONF=*confname*	Assign a configuration to another host
&CLOSE DISTRIBUTION	End UDS-D mode
COMMIT *transaction-id*	Terminate transaction in PTC state by committing updates (FINISH) without taking cross-configuration consistency or UDS/SQL and openUTM consistency into account
&DISPLAY DISTRIBUTION [,NODE=*processor-name*] [,CONF=*confname*] [,DB=*dbname*] [,SS=*subschema-name*]	Display distribution table
&DROP DISTRIBUTION, {ALL | NODE=*processor-name*, CONF=*confname*[,DB=*dbname*][,ALL] | DB=*dbname*,SS=*subschema-name* }	Delete entries in the distribution table

&LOCK DISTRIBUTION, {NODE=*processor-name* | CONF=*confname* | DB=*dbname* | SS=*subschema-name* }	Lock entries in distribution table
MODIFY PTCSYNCH, VALUE=([{WAIT | ABORT | COMMIT}] [,[{WAIT | ABORT | COMMIT}]])	Modify value of the DBH load parameter PTCSYNCH
&PWD DISTRIBUTION,CONF=*confname*, {PWN=*new-password* | PWO=*old-password*| PWO=*old-password*,PWN=*new-password*}	Assign and modify password
&SAVE DISTRIBUTION,FILE=file*name*	Save distribution table
&START DISTRIBUTION	UDS-D mode starten
&SYNCHRONIZE DISTRIBUTION	Terminate secondary subtransactions in PTC state
&UNLOCK DISTRIBUTION, {NODE=*processor-name* | CONF=*confname* | DB=*dbname* | SS=*subschema-name* }	Unlock entries in the distribution table

Table 34: DAL commands for UDS-D

The DAL commands are described in detail in section “The Database Administrator Language DAL”.

15 Appendix

- Function codes of the DML statements

15.1 Function codes of the DML statements

The following table shows the function codes in decimal notation.

Function code	Short function name	Meaning
0	START	Start a user program
1	FINISH	FINISH
2	ERASUN	ERASE *record-name*
3	ERASQU	ERASE *record-name* {PERMANENT/SELECTIVE/ALL}
4	FND1	FIND-1
5	FND5	FIND-5
6	FND4R	FIND-4 without WITHIN or with WITHIN *realm-name*
7	FND4S	FIND-4 WITHIN *set-name*
8	FND6	FIND-6
9	FND2	FIND-2
10	FND7IS	FIND-7 USING *record-element-name*,... (sequential search)
11	FND7IK	FIND-7 USING *record-element-name*,... (search with key)
12	FND3IS	FIND-3 USING *record-element-name*,... (sequential search)
13	FND3IK	FIND-3 USING *record-element-name*,... (search with key)
15	CONNEC	CONNECT
16	MODIFY	MODIFY
17	ACCEPT	ACCEPT
18	READY	READY
19	DISCON	DISCONNECT
20	STORE	STORE
21	STOP	End a user program
22	FREE	FREE
23	KEEP	KEEP
24	IF_	IF

25	FND7SE	FIND-7 without USING or with USING *search-expression*
26	ONLINE	UDS online utility
28	FND3SE	FIND-3 without USING
30	LONAM	LOOK for a name
31	LOREC	LOOK for a record
32	LOSET	LOOK for a set
33	LOITM	LOOK for an item
34	LOKEY	LOOK for a key
36	LORLM	LOOK for a realm
37	RDYGLO	Start a distributed transaction[1]
38	PTC	End initiation phase for distributed transaction[1]
125	RELBAS	RELOCATE RELOCATION-TYPE = *BASE-LEVEL-TABLE-PAGES
126	RELINX	RELOCATE RELOCATION-TYPE = *INDEX-LEVEL-TABLE-PAGES
127	RELDST	RELOCATE RELOCATION-TYPE = *DISTRIBUTABLE-TABLE-PAGES
128	RELREC	RELOCATE RELOCATION-TYPE = *RECORD-PAGES
129	FPASCA	FPASCAN
130	PRERLM	PREFRLM
131	REOPPP	REORGPPP
132	FTC1	FETCH-1
133	FTC5	FETCH-5
134	FTC4R	FETCH-4 without WITHIN or with WITHIN *realmname*
135	FTC4S	FETCH-4 with WITHIN *setname*
136	FTC6	FETCH-6
137	FTC2	FETCH-2
138	FTC7IS	FETCH-7 USING *record-element-name*,... (sequential search)
139	FTC7IK	FETCH-7 USING *record-element-name*,... (search with key)
140	FTC3IS	FETCH-3 USING *record-element-name*,... (sequential search)

141	FTC3IK	FETCH-3 USING *record-element-name*,... (search with key)
142	GET	GET
153	FTC7SE	FETCH-7 without USING or with USING *search-expression*
156	FTC3SE	FETCH-3 without USING
158	NLONAM	next LOOK for a name
160	NLOSET	next LOOK for a set
161	NLOITM	next LOOK for an item
162	NLOKEY	next LOOK for a key
164	NLORLM	next LOOK for a realm

Table 35: Function codes of the DML statements

[1] Only in conjunction with UDS-D

16 Glossary

This Glossary contains the definitions of some of the important terms and concepts used in the UDS/SQL manuals. Terms that appear in *italics* within a particular definition have also been defined in this Glossary.
In cases where two or more terms are used synonymously, a “See” reference points to the more commonly used term in these manuals.

A

access, contending

See *contending access*.

access, direct

See *direct access*.

access, sequential

See *sequential access*.

access authorization

The rights of a specified user group with regard to access to the *database* . Access rights are defined during live database operation using ONLINE- PRIVACY utility routine or, in offline mode, using the BPRIVACY utility routine.

access path

Means of finding a certain subset of all *records* qualified by a search query, without having to carry out a sequential search of the whole *database*.

access rights

Right of access to a *database* as defined in the BPRIVACY utility routine.

access type

Type of access, e.g. read, update etc.

act-key

(actual key) Actual address of a *page*, consisting of *realm number* and *page number*.

act-key-0 page

First *page* of a *realm*; contains general information on the realm such as

- when the realm was created,
- when the realm was last updated,
- *internal version number* of the realm,
- *system break information*
- if applicable, *warm start* information.

act-key-N page

Characteristic page of a *realm*, with the highest *page number*. Copy of the *act-key-0 page*.

address, physical

See *act-key* or *probable position pointer (PPP)*.

administrator task

Task of the *independent DBH*; The *database administrator* can control execution of the *independent DBH* via this task.

AFIM

See *after-image*.

after-image

Modified portion of a *page* **after** its content has been updated. The *DBH* writes after-images to the *RLOG file* as well as the *ALOG file*.

after-image, ALOG file

The after-images are written to the ALOG file when the ALOG buffer is full. The purpose of the after-images in the ALOG file is to secure the data contained in the database and thus they must be maintained for a long period of time. They are used to reconstruct an original database or update a *shadow database*.

after-image, RLOG file

After-images are logged in the RLOG file **before** the updates are applied to the *database*. The after-images held in the RLOG file are required for *warm start* only. They are thus periodically overwritten.

ALOG file

File for securing the data contained in the database in the long term; see *after-image*.

ALOG sequence number

See *sequence number*.

anchor record

Record automatically created by UDS/SQL as *owner record* for *SYSTEM sets*. It cannot contain any *items* defined with the *schema DDL* and cannot be accessed.

application

Realization of a job in one or several *user programs* working with UDS/SQL *databases*.

application program (AP)

E.g. *COBOL DML* program or IQS.

area

See *realm*.

ascending key (ASC key)

Primary key of a *set*. Defines the sequence of *member records* in the *set occurrences* by ascending key values.

authorization

Identification used for user groups.

authorized users

Specified user groups who are authorized to access the *database*.

automatic DBTT extension

Some utility routines automatically extend the number of records possible for a record type if too few are available; no separate administration is required to do this.
See also *online DBTT extension*.

automatic realm extension

Some utility routines automatically extend realms when insufficient free space is available; no separate administration is required to do this. See also *online realm extension*.

B

backup database

See *shadow database*.

base interface block (BIB)

(Base Interface Block) Standard interface between UDS/SQL and each individual user; it contains, among other things, the *RECORD AREA* (user records as defined in the *subschema*).

before-image

Copy of a *page* taken before its contents are updated.

The *DBH* writes before-images to the *RLOG files* during database operation before the updates are applied to the *database*. A prerequisite is that the RLOG files exist.

BFIM

See *before-image*.

BIB

See *base interface block*.

buffer pool

See *system buffer pools* and *exclusive buffer pool*.

C

CALC key

Key whose value is converted into a relative *page number* by means of a *hash routine.*

CALC page

Page of a *hash area*.

CALC SEARCH key

Secondary key. Used as *access path* for *direct access* via *hash routine*.

CALC table

Table in the direct/indirect *CALC page* whose entries point to the stored records. Each line contains:

- the *CALC key*,
- the *record sequence number*
- the displacement to the related *page index entry* (direct CALC page) or the *probable position pointer* (indirect CALC page).

CALL DML

DML that is called by various programming languages (Assembler, COBOL, FORTRAN, PASCAL, PL/1) via the CALL interface.

catalog identifier

Name of the public volume set (PVS) under which the BS2000 UDS/SQL files are stored. The catalog identifier is part of the database or file name and must be enclosed in colons: “:*catid*:”.

chain

Storage mode for a *set occurrence* in which every *record* contains a pointer to the subsequent record.

Character Separated Values (CSV)

Output format in which the values are separated by a predefined character.

checkpoint

Consistency point , at which the ALOG file was changed and to which it is possible to return at any time using BMEND utility routine.

check records

Elements which provide information for checking the database. They vary in length from 20 to 271 bytes.

CHECK-TABLE

Check table produced by the *DDL* compiler during *Subschema DDL* compilation, and used by the COBOL compiler and *CALL DML* to check whether the *DML* statements specified in the *application program* are permitted. It is part of the *COSSD* or *SSITAB module* .

clone pair, clone pubset, clone session, clone unit

A clone unit is the copy of an (original) unit (logical disk in BS2000) at a particular time (“Point-in-Time copy”). The TimeFinder/Clone component creates this copy optionally as a complete copy or as a “snapshot”.
After they have been activated, the unit and clone unit are split; applications can access both.
The unit and clone unit together form a clone pair. TimeFinder/Clone manages this pair in what is known as a clone session.
If clone units exist for all units of a pubset, these clone units together form the clone pubset.
Details of this are provided in the manual "Introduction to System Administration" .

COBOL DML

DML integrated in the COBOL language.

COBOL runtime system

Runtime system; sharable routines selected by the COBOL compiler (COBOL2000 or COBOL85) for the execution of complex statements.

COBOL Subschema Directory (COSSD)

Provides the COBOL compiler with subschema information for compilation of the DB *application programs* .

common memory

Shareable memory area used by several different tasks. In UDS/SQL, it always consists of the *common pool* and the *communication pool* and, depending on the application, the SSITAB pool (see *SSITAB module*) if *CALL DML* is used.
If UDS-D is used, it also consists of the *distribution pool* and the *transfer pool* .

common pool

Communication area of the *independent DBH* . Enables *DBH* modules to communicate with each other. Contains, among other things, an input/output buffer for *pages* (*buffer pools*).

communication partners

Tasks or data display terminals.

communication pool

Communication area of the *independent DBH* for *application programs* . One of its functions is to store base interface blocks (*BIB*).

compatible database interface (KDBS)

see *KDBS.*

compiler database

The *realms* and files of the *database* which are required by the UDS/SQL compiler. They are

- *DBDIR* (*Database Directory*)
- *DBCOM* (*Database Compiler Realm*)
- *COSSD* (*COBOL Subschema Directory*).

COMPILER-SCHEMA

UDS/SQL-internal *schema* of the *compiler database*.

COMPILER-SUBSCHEMA

UDS/SQL-internal *subschema* of the *compiler database*.

compound key

Key consisting of several *key items*.

compression

Only the filled *items* of a *record* are stored (see *SSL* clause COMPRESSION).

configuration

See *DB configuration*.

configuration user ID

User ID in which the *database administrator* starts the *DBH*.

configuration name

Freely selectable name of the *database configuration* for a particular *session*. The *DBH* uses it to form:

- the name of the *Session Log File*,
- the names of the *DB status file* and its backup copy,
- the names of the *RLOG files*,
- the names of the temporary *realms*,
- the names of session job variables,
- the *event names* of *P1 eventing*,
- the *DCAM application* name for the administration,
- the names of the *common pools*
- the names of the dump files.

connection module

Module that must be linked into every UDS/SQL *application program* and which establishes the connection with the *DBH*.

consistency

State of the database without conflicts in the data stored in it.

consistency, logical

State of the database in which the stored data has no internal conflicts and reflects the real-world situation.

consistency, physical

State of the database in which the stored data is consistent with regard to correct physical storage, *access paths* and description information.

consistency, storage

See *physical consistency*.

consistency error

A violation of the *physical consistency* of the stored data.

consistency point

Point (in time) at which the *database* is consistent, i.e. all modifying transaction have been terminated and their modifications have been executed in the database.

consistency record

Administration record with consistency time and date stamps in the *DBDIR*. For an update in a *realm* the *DBH* enters the date and time in the consistency record and in the updated realm. When realms or *databases* are attached for a *session*, the DBH uses this time stamp to check the consistency of the realms within each database.

contending access

Different *transactions* attempting to access a *page* simultaneously.

conversation

SQL -specific administration data is retained across transaction boundaries in an *SQL* application. This kind of data administration unit is called a conversation.
In openUTM such an administrative unit is also called a service.

copy

See *database copy*.

COSSD

See *COBOL Subschema Directory*.

CRA

(Current Record of Area) *Record* which is marked in the *currency table* as the current record of a particular *realm* (area).

CRR

(Current Record of Record) *Record* which is marked in the *currency table* as the current record of a particular *record type* (Record) .

CRS

(Current Record of Set) *Record* which is marked in the *currency table* as the current record of a particular *set*.

CRU

(Current Record of Rununit) *Record* which is marked in the *currency table* as the current record of the *processing chain*.

CSV

see *Character Separated Values.*

currency table

The currency table contains:

- CURRENT OF AREA table (table of CRAs),
- CURRENT OF RECORD table (table of CRRs) and
- CURRENT OF SET table (table of CRSs).

CURRENT OF AREA table

See *currency table*.

CURRENT OF RECORD table

See *currency table*.

CURRENT OF SET table

See *currency table*.

D

DAL

(Database Administrator Language) Comprises the commands which monitor and control a *session*.

data backup

Protection against loss of data as a result of hardware or software failure.

data deadlock

See *deadlock.*

data protection (privacy)

Protection against unauthorized access to data. Implemented in UDS/SQL by means of the schema/subschema concept and access authorization. *Access rights* are granted by means of the BPRIVACY utility routine.

database (DB)

Related data resources that are evaluated, processed and administered with the help of a *database system*.
A database is identified by the database name. An UDS/SQL database consists of the *user database* and the *compiler database*.
To prevent the loss of data, a *shadow database* may be operated together with (i.e. parallel to) the original database.

database administrator

Person who manages and controls *database* operation. The DB administrator is responsible for the utility routines and the Database Administrator Language (*DAL*).

database copy

Copy of a consistent *database*; may be taken at a freely selectable point in time.

database compiler realm (DBCOM)

Stores information on the *realms*, *records* and *sets* defined by the user in the *Schema DDL* and *Subschema DDL*.

database copy update

Updating of a *database copy* to the status of a *checkpoint* by applying the appropriate *after-images*.

database directory (DBDIR)

Contains, among other things, the *SIA*, all the *SSIAs* and information on *access rights*.

database job variable

Job variable in which UDS/SQL stores information on the status of a *database*.

database key (DB key)

Key whose value represents a unique identifier of a *record* in the *database*. It consists of the *record reference number* and the *record sequence number*. The database key values are either defined by the database programmer or automatically assigned by UDS/SQL.

database key item

Item of type DATABASE-KEY or DATABASE-KEY-LONG that is used to accommodate *database key* values. Items of type DATABASE-KEY and DATABASE-KEY-LONG differ in terms of the item length (4 bytes / 8 bytes) and value range.

DATABASE-KEY item

See *database key item*.

DATABASE-KEY-LONG item

See *database key item*.

database page

See *page*.

DATABASE-STATUS

Five-byte item indicating the database status and consisting of the *statement code* and the *status code*.

database system

Software system that supports all tasks in connection with managing and controlling large data resources. The database system provides mechanisms for stable and expandable data organization without redundancies. They allow many users to access *databases* concurrently and guarantee a consistent data repository.

DB status file

(database status file) Contains information on the most recently reset *transactions*. openUTM-S or, in the case of distributed processing, UDS-D/openUTM-D needs this information for a *session restart*.

DB configuration

(database configuration) The *databases* attached to a *DBH* at any one point during *session* runtime. As the result of *DAL* commands or DBH error handling, the database configuration can change in the course of a session. At the *session start*, the DB configuration may be empty. Databases can be attached with DAL commands after the start of the session. They can also be detached during the session with DAL commands.

DBCOM

See *database compiler realm*.

DBDIR

See *database directory*.

DBH

Database Handler: program (or group of programs) which controls access to the *database(s)* of a *session* and assumes all the attendant administrative functions.

DBH end

End of the *DBH* program run. DBH end can be either a *session end* or a *session abort*.

DBH, independent

See *independent DBH*.

DB key

See *database key*.

DBH, linked-in

See *linked-in DBH*.

DBH load parameters

See *load parameters* (*DBH*).

DBH start

Start of the *DBH* program run. DBH start can be either a *session start* or a *session restart*.

DBTT

(Database Key Translation Table) Table from which UDS/SQL can obtain the *page address* (*act-key*) of a *record* and associated tables by means of the database key value.

The DBTT for the SSIA-RECORD consists only of the DBTT base. For all other record types, the DBTT consists of a base table (DBTT base) and possibly of one or more extension tables (DBTT extents) resulting from an online DBTT extension or created by BREORG.

DBTT anchor page

Page lying within the realm of the associated DBTT in which the DBTT base and DBTT extents are administered. Depending on the number of DBTT extents multiple chained DBTT anchor pages may be required for their administration.

DBTT base

see *DBTT*

DBTT extent

see *DBTT*

DBTT page

Page containing the *DBTT* or part of the *DBTT* for a particular *record type*.

DCAM

Component of the TRANSDATA data communication program.

DCAM application

Communication application using the *DCAM* communication method. A DCAM application enables communication between

- a DCAM application and terminals.
- different DCAM applications within the same or different hosts, and with *remote configurations*.
- a DCAM and a openUTM application.

DDL

(Data Description Language) Formalized language for defining the logical data structure.

deadlock

Mutual blocking of *transactions*. A deadlock can occur in the following situations:

- Data deadlock: This occurs when *transactions* block each other with *contending access*.
- Task deadlock: This occurs when a *transaction* that is holding a lock cannot release it, since no openUTM task is free. This deadlock situation can only occur with UDS/SQL-openUTM interoperation.

descending key (DESC key)

Primary key of a set. Determines the sequence of *member records* in the *set occurrences* to reflect descending key values.

direct access

Access to a *record* via an item content. UDS/SQL supports direct access via the *database key, hash routines* and *multi-level tables*.

direct hash area

See *hash area*.

distributed database

A logically connected set of data resources that is distributed over more than one UDS/SQL configuration.

distributed transaction

Transaction that addresses at least one *remote configuration*. A transaction can be distributed over:

- UDS-D,
- openUTM-D,
- UDS-D and openUTM-D.

distribution pool

Area in the *independent DBH* used for communication between *UDSCT*, *server tasks*, *user tasks* and the *master task* with regard to UDS-D-specific data. The distribution pool contains the *distribution table* and the UDS-D-specific system tables.

distribution table

Table created by UDS-D using the input file assigned in the *distribution pool*. With the aid of the distribution table, the distribution component in the *user task* decides whether a *processing chain* should be processed locally or remotely. Assigned in the distribution table are:

subschema - *database*

database - *configuration*

configuration - host computer.

DML

Data Manipulation Language: language for accessing a UDS/SQL *database*.

dummy subtransaction

A primary *subtransaction* is created by UDS-D when the first *READY* statement in a *transaction* addresses a *remote database* .
A dummy subtransaction is used to inform the *local configuration* of the transaction so that the *database* can be recovered following an error.

duplicates header

Contains general information on a *duplicates table* or a *page* of a duplicates table, i.e.

- chaining reference to the next and previous *overflow page*
- the number of free bytes in the page of the duplicates table.

duplicates table

Special *SEARCH-KEY table* in which a key value which occurs more than once is stored only once.
For each key value, the duplicates table contains:

- a table index entry with the key value and a pointer to the associated table entry
- a table entry (DB key list), which can extend over several pages, containing the *record sequence numbers* of the *records* which contain this key value.

duplicates table, main level

Main level, Level 0. Contains a table index entry and the beginning of the associated table entry (DB key list).

dynamic set

Set which exists only for the life of a *transaction* and which stores *member records* retrieved as result of search queries.

E

ESTIMATE-REPORT

Report produced after BGSIA run. Used to estimate the size of the *user realms*.

event name

Identification used in eventing.

exclusive buffer pool

Buffer which, in addition to the *system buffer pools*, is used exclusively for buffering *pages* of the specified *database*.

F

foreign key

Record element whose value matches the primary key values of another table (UDS/SQL *record type*). Foreign keys in the sense of UDS/SQL are qualified as "REFERENCES owner record type" in the member record type of a set relationship in the BPSQLSIA protocol.

FPA

See *free place administration*.

FPA base

See *free place administration*.

FPA extent

See *free place administration*.

FPA page

Free place administration page.

free place administration (FPA)

Free space is managed both at realm level (*FPA pages*) and at page and table level. Free place administration of the pages is carried out in a base table (FPA base) and possibly in one or more extension tables (FPA extents) created by means of an online realm extension or BREORG.

function code

Coding of a *DML* statement; included in information output by means of the *DAL* command DISPLAY or by UDSMON.

G

group item

Nameable grouping of *record elements*.

H

hash area

Storage area in which UDS/SQL stores data and from which it retrieves data on the basis of key values which are converted into relative *page numbers*. A hash area may contain the *record* addresses as well as the records themselves. A *direct hash area* contains the records themselves; an *indirect hash area*, by contrast, contains the addresses of records stored at some other location.

hash routine

Module which performs *hashing*.

hashing

Method of converting a key value into a *page address*.

HASHLIB

Module library for the storage of *hash routines* for one *database*.

I

identifier

Name allocated by the database designer to an *item* that UDS/SQL creates automatically. UDS /SQL adapts item type and length to the specified item usage.

implicit set

SYSTEM set created by UDS/SQL when a *SEARCH key* is defined at record type level.

inconsistency

State of the database in which the data values contained in it are inconsistent.

independent DBH

Independent program system enabling more than one user to access a single *database* (*mono-DB operation*) or several databases (*multi-DB operation*) simultaneously. The independent DBH is designed as a task family, consisting of

- a *master task* (*UDSSQL*)
- one or more *server tasks* (*UDSSUB*)
- an *administrator task* (*UDSADM*)

index level

Hierarchy level of an *index page*.

index page

Page in which the highest (lowest) key values of the next-lower level of an indexed table are stored.

INDEX search key

Secondary key. Used as *access path* for *direct access* via a *multi-level table*.

indirect hash area

See *hash area*.

integrity

State of the database in which the data contained in it is complete and free of errors.

- entity integrity
- *referential integrity*
- user integrity

interconfiguration

Concerning at least one *remote configuration*.

interconfiguration consistency

A *distributed transaction* that has caused updates in at least one *remote configuration* is terminated in such a way that the updates are either executed on the *databases* in each participating *DB configuration* or on none at all.
Interconfiguration consistency is assured by the *two-phase commit protocol*.

interconfiguration deadlock

Situation where *distributed transactions* are mutually locked due to *contending accesses*.

interface

In software: memory area used by several different programs for the transfer of data.

internal version number

Each *realm* of the *database*, including *DBDIR* and *DBCOM*, has an internal version number which the utility routines (e.g. BREORG, BALTER) increment by one whenever a realm is updated. This internal version number is kept in the *act-key-0 page* of the realm itself and also in the PHYS VERSION RECORD in the DBDIR.

item

Smallest nameable unit of data within a *record type*. It is defined by item type and item length.

K

KDBS

Compatible database interface. Enables programs to be applied to applications of *DB systems* by different manufacturers.

key

Item used by the database programmer for *direct access* to records; an optimized *access path* is provided for the key by UDS/SQL in accordance with the *schema* definition.

key, compound

Key consisting of several *key items*.

key item

Item defined as a *key* in the *schema*.

key reference number

Keys are numbered consecutively in ascending order, beginning at 1.

L

linked-in control system

UDS/SQL component for *linked-in DBH*, responsible for control functions (corresponds to the *subcontrol system* of the *independent DBH*).

linked-in DBH

Module linked in to or dynamically loaded for the current DB *application program* and controlling access to a single *database* (*mono-DB operation*) or several databases simultaneously (*multi-DB operation*).

list

Table containing the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

In a distributable list the data pages which contain the member records (level 0 pages) can be distributed over more than one realm. The pages containing the higher-ranking table levels all reside in one realm (table realm of a distributable list).

load parameters (DBH)

Parameters requested by the *DBH* at the beginning of the *session*. They define the basic characteristics of a session.

local application program

An *application program* is local with regard to a *configuration* if it was linked to the configuration using

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILENAME=conf-name
```

local configuration

The *configuration* assigned to an *application program* before it is called using

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name
```

The application program communicates with the local configuration via the *communication pool*. The local configuration is in the same host as the application program.

local database

Database in a *local configuration*.

local distribution table

A *distribution table* is considered local to a *DBH* if it is held in the DBH's *distribution pool*.

local host

Host computer containing the *application program*.

local transaction

Transaction that only addresses the *local configuration*.

logging

Recording of all updates in the *database*.

logical connection

Assignment of two *communication partners* that enables them to exchange data.
DCAM applications communicate via logical connections.

M

main reference

In the *DBH* the main reference is used to manage the resources required for processing a transaction's requests, including those for transferring the requests from the application program to the DBH and back.

mainref number

Number assigned to the *transaction* at *READY*. This number is unique only at a given time; at the end of the transaction, it is assigned to another transaction.

master task

Task of the *independent DBH* in which the *UDSQL* module executes. Controls the start and end of a *session* and communicates with the *database administrator* directly or via the *administrator task*.

member

See *member record* or *member record type*.

member, AUTOMATIC

Record is inserted at storage time.

member, MANDATORY

Record cannot be removed.

member, MANUAL

Record is not inserted automatically at storage time.

member, OPTIONAL

Record can be removed.

member record

Lower-ranking *record* in a *set occurrence*.

member record type

Lower-ranking *record type* in a *set*.

mono-DB configuration

Type of configuration where only one *database* takes part in a *session*.

mono-DB operation

Mode of *database* operation where the *DBH* uses only one database of a *configuration*.

multi-DB configuration

Type of configuration where several *databases* take part in a *session*.

multi-DB operation

Mode of database operation where the *DBH* uses several *databases* of a *configuration*.

multi-DB program

Application program that addresses more than one *database*. The databases may be part of one or more *mono-DB* or *multi-DB configurations*.

multi-level table

SEARCH KEY table which contains a line for each *record* of the associated *record type* or each *member record* of the *set occurrence*, as appropriate. Each line comprises the key value of the record and the record pointer. It is also referred to as an indexed table.

multithreading

A mechanism that enables the *DBH* to fully exploit the CPU. Multithreading means that the DBH processes several jobs concurrently by using so-called threads. Each thread has information on the current status of a particular job stored in it. When a job needs to wait for the completion of an I/O operation, DBH uses the CPU to process some other job.

N

network

All computers linked via TRANSDATA.

O

OLTP

(Online Transaction Processing) In an OLTP application, a very large number of users access the same programs and data. This usually occurs under the control of a transaction monitor (TP monitor).

online backup

If AFIM logging is active, the *database* can be saved during a session. The ability to save a database online is determined with the BMEND utility routine.

online DBTT extension

Extension during ongoing database operation of the number of possible records of a record type. The DAL commands ACT DBTT-INCR, DEACT DBTT-INCR, DISPLAY DBTT-INCR and EXTEND DBTT can be used to administer the online extension of DBTTs.

See also *automatic DBTT extension*.

online realm extension

Extension of *user realms* and *DBDIR* in ongoing database operation. The DAL commands ACT INCR, DEACT INCR, DISPLAY INCR, EXTEND REALM and REACT INCR are provided for administering the online extensibility of realms.

See also *automatic realm extension*.

open transaction

Transaction which has not been closed with FINISH or FINISH WITH CANCEL, or with COMMIT or ROLLBACK.

openUTM

(universal transaction monitor) Facilitates the creation and operation of transaction-oriented applications.

operator task (OT)

See *master task*

original database

The term “original database” refers solely to the naming of the database files (*dbname.dbfile*), not to the status of the database content (see also *shadow database*).

overflow page

Page in *hash areas* and *duplicates tables* for storing data that does not fit in the primary page. Their structure is the same as that of the pages of the hash area or duplicates table in question.

owner

See *owner record* or *owner record type*.

owner record

Higher-ranking *record* in a *set occurrence*.

owner record type

Higher-ranking *record type* in a *set*.

P

page

Physical subunit of a *realm*. UDS/SQL identifies pages by means of unique keys (*act-key*).

The length of a page may be optionally 2048, 4000 or 8096 bytes. All pages within a database must have the same length. Pages with a length of 4000 or 8096 bytes are embedded in a *page container*.

page address

In a page address, a distinction is made between the current address of a *page*, i.e. the *act-key*, and the probable address of a page, the *probable position pointer (PPP)*.

page container

Pages with a length of 4000 or 8096 bytes are embedded in a so-called page container, which consists of a 64-byte header that precedes the page and a 32-byte trailer at the end of the page.

page header (page info)

The first 20 bytes of a database *page* (except for the *FPA* and *DBTT pages* with a length of 2048 bytes). They contain:

- the *act-key* of the *page* itself,
- the number of *page index entries*
- the length and displacement of the bytes which are still vacant in this page.
- the page type (*ACT-Key-0 page*, *FPA page*, *DBTT page*, *DBTT anchor page*, normal data page or *CALC page*)

page index entry

Indicates the position of a *record* within a *page.*

page number

In each *realm* the *pages* are numbered consecutively in ascending order starting starting from 0. The page number is part of the *page address*.
Page number = PAM page number -1 for databases with a page length of 2048 bytes
Page number = (PAM page number-1) / 2 for databases with a page length of 4000 bytes
Page number = (PAM page number-1) / 4 for databases with a page length of 8096 bytes.

password for UDS/SQL files

Password serving to protect the files created by UDS/SQL (default: C'UDS'BLANK''). The *DB administrator* can define other passwords with PP CATPASS or MODIFY-FILE-ATTRIBUTES.

pattern

Symbolic representation of all possible *item* contents, used at item definition.

pattern string

String defining a *pattern*.

PETA

Preliminary end of transaction: UDS-D or openUTM-D statement that causes a preliminary transaction end.
The PETA statement belongs to the first phase of the *two-phase commit protocol* which terminates a *distributed transaction*.
The PETA statement stores the following information failproof in the *RLOG file* of the local *DBH*:

- each updated *page*
- rollback and locking information
- the names of all participating *configurations*.

This information is required for any future *warm start*.

pointer array

Table of pointers to the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

PPP

See *probable position pointer (PPP)*.

prepared to commit (PTC)

Part of the *two-phase commit protocol*:
State of a *subtransaction* after execution of a *PETA* statement and before receipt of the message that the complete *transaction* is to be terminated with FINISH or FINISH WITH CANCEL.

primary key

Distinguished from *secondary keys* for reasons of efficiency. Usually a unique identifier for a *record*.

primary key (DDL)

The *key* of a *record type* which is defined by means of "LOCATION MODE IS CALC" or the *key* of an order-determining *key* of a set occurrence which is defined by means of "ORDER IS SORTED [INDEXED]". Also used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

primary key (SQL)

In the broader sense (SQL), a *record element* uniquely identifying a record. In UDS-SQL, the database key of an owner record output as the "PRIMARY KEY" in the BPSQLSIA log (see also *foreign key*). A *record element* which uniquely identifies a record is flagged as "UNIQUE" in the BPSQLSIA log unless it is the aforementioned "PRIMARY KEY".

primary subtransaction

Subtransaction that runs in the *local configuration*. The primary subtransaction is opened by the first *READY* statement in a *transaction* on a *local database*. If the first READY statement addresses a *remote database*, UDS-D generates a *dummy subtransaction* as the primary subtransaction.

PRIVACY-AND-IQF SCHEMA

UDS/SQL-internal *schema* for protection against unauthorized access.

PRIVACY-AND-IQF SUBSCHEMA

UDS/SQL-internal *subschema* for protection against unauthorized access.

probable position pointer (PPP)

Probable address of a *page*, comprising *realm number* and *page number*. UDS/SQL does not always update probable position pointers (PPP) when the storage location of data is changed.

processing chain

Sequence of DML statements applied to a *database* within a *transaction*.

PTC state

See *prepared to commit*.

pubset declaration

Siee *UDS/SQL pubset declaration*

pubset declaration job variable

Job variable in which a *UDS/SQL pubset declaration* is specified.

P1 eventing

Manner in which tasks communicate with each other.

R

READY

Start of a *transaction* or a *processing chain* in *COBOL DML* programs.

READYC

Start of a *transaction* or a *processing chain* in *CALL DML* programs.

realm

Nameable physical subunit of the *database*. Equivalent to a file. Apart from the *user realms* for user data there are also the realms *DBDIR* and *DBCOM*, which are required by UDS/SQL.

realm configuration

Comprises all the database *realms* taking part in a *session*.

realm copy

See *database copy*.

realm reference number

Realms are numbered consecutively in ascending order, starting with 1. The realm reference number (area reference) is part of the *page address*.

reconfiguration

Regrouping of databases in a *DB configuration* after a *session abort*. A prerequisite for reconfiguration is that the *SLF* has been deleted or that its contents have been marked as invalid.

record

Single occurrence of a *record type*; consists of one item content for each of the *items* defined for the record type and is the smallest unit of data managed by UDS/SQL via a unique identifier, the *database key*. The reserved word RECORD is used in DDL and SSL syntax to declare a record type.

record address

Address of the page containing the *record*. See *page address*.

RECORD AREA

Area in the *USER WORK AREA* (*UWA*) which can be referenced by the user. The record area contains the *record types* and the implicitly defined items (IMPLICITLY-DEFINED-DATA-NAMES) of the database such as the AREA-ID items of the WITHIN clauses of the schema. The length of the record area is essentially defined by the record types contained in it.

record element

Item, *vector* or *group item*.

record hierarchy

Owner/member relationship between *record types*:
the *owner record type* is the higher-ranking part of the relationship;
the *member record type* is the lower-ranking part.

REC-REF

See *record reference number*.

record reference number

Record types are numbered consecutively in ascending order, starting at 1. The record reference number is part of the *database key*.

record SEARCH KEY table

SEARCH KEY table for selection of a *record* from a *record type*.

record sequence number (RSQ)

The record sequence number can be assigned by the database programmer; if not, UDS/SQL numbers the *records* of a *record type* contiguously in ascending order, in the sequence in which they are stored; numbering starts at 1. The record sequence number is part of the *database key*.

record type

Nameable grouping of *record elements*.

record type, linear

Record type that is neither the *owner* nor the *member* of a set (corresponds to record types of a conventional file).

referential integrity

Integrity of the relationships between tables (UDS/SQL *record types*).

remote application program

Application program that is not local with regard to a particular *configuration*.

remote configuration

DB-configurations that are not assigned to the *application program* via /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name but via the *distribution table* once the application program is running. The *connection module* of the application program communicates with the remote configurations via *DCAM applications*.
Remote configurations can be situated on *local* or *remote* hosts.

remote database

Database in a *remote configuration*.

remote host

Host computer that is not local.

repeating group

Group item with repetition factor. The repetition factor, which must be greater than 1, specifies the number of duplicates of the group item to be incorporated in the repeating group.

request

The functions of the *DAL* commands ADD DB, ADD RN, DROP DB, DROP RN, NEW RLOG and CHECKPOINT are held in the *DBH* as "requests" and are not executed until the DAL command PERFORM is entered.

restart of BMEND

Resumption of an aborted BMEND run.

restart of a session

See *session restart*.

restructuring

Modification of the *Schema DDL* or *SSL* for *databases* already containing data.

return code

Internal code which the called program sends to the calling program;
Return code != 0 means an error has occurred.

RLOG file

Backup file used by the *DBH* during a session to store *before-images* (BFIMs) and *after-images* (AFIMs) of data which is updated. With the aid of the *RLOG file*, the DBH can cancel updates effected by incomplete *transactions*. There is one RLOG file per *configuration*. An RLOG file consists of two physical files.

rollback

Canceling of all updates effected within a *transaction*.

RSQ

See *record sequence number*.

RUNUNIT-ID

See *transaction identification.*

S

schema

Formalized description of all data structures permitted in the *database*. A UDS/SQL schema is defined by means of the *Schema DDL*.

Schema DDL

Formalized language for defining a *schema*.

Schema Information Area (SIA)

The SIA contains the complete database definition. The *DBH* loads the SIA into main memory at the start of DB processing.

SEARCH KEY

Secondary key; *access paths* using secondary keys are created by UDS/SQL by means of *hash routines* and *multi-level tables*.

SEARCH KEY table

Multi-level table used by UDS/SQL as an *access path* via a *secondary key*.

secondary key

Any *key* which is not a *primary key*. Used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

secondary subtransactions

Subtransactions that address *remote configurations*.

sequence number

Identifier in the name of the *ALOG files* (000000001 - 999999999). The first ALOG file of a *database* is always numbered 000000001.

sequential access

Accessing a *record* on the basis of its position within a predefined record sequence.

server task

Task of the *independent DBH* in which the *UDSSUB* module executes; processes the requests of the DB *application programs*.

session

Period between starting and normal termination of the *DBH* (*independent* / *linked- in*) in which it is possible to work with the *databases* of the *configuration*. Normally, a session consists of a sequence of *session sections* and *session interrupts.*

session abort

Occurs when the *DBH* is terminated abnormally after a successful *session start*. A session abort can be caused by: power failure, computer failure, BS2000 problems, DBH problems, %TERM.

session end

Is the result of:

- *DAL* when using *independent DBH*,
- TERM in the *DML application program* when using *linked-in DBH*,
- *DBH* error handling.

During a *session interrupt*, the user can also effect session end by invalidating the *SLF* contents. Inconsistent *databases* can be made consistent again by a *warm start*, even without an SLF.

session interrupt

The period between a *session abort* and the related *session restart*.

session job variable

Job variable in which UDS/SQL stores information about a session.

Session Log File (SLF)

File which is permanently assigned to a *session* and which is required by the *DBH* in the event of a *session restart*. It contains information on the current DB *configuration*, the number of current file identifiers and the current values of the *DBH load parameters*.

session restart

Starting of the *DBH*, under the same *configuration name* and *configuration user ID*, after a *session abort*. With the aid of the *SLF*, the *DBH load parameters* and the current file identifiers which existed when the session aborted are re-established, and the *databases* of the previous *configuration* are reattached, if necessary by means of a *warm start*.

session section

Period from the start of the *DBH*, either at the *session start* or a *restart*, to the normal *session end* or to a *session abort*.

session section number

Number which identifies a session section unambiguously.

session start

State of a session in which the *DBH* is started under a configuration name for which there is no *Session Log File* (*SLF*) with valid contents.

set

Nameable relationship between two *record types*.

set, dynamic

See *dynamic set*.

set, implicit

See *implicit set*.

set, singular

See *SYSTEM set*.

set, standard

See *standard set*.

Set Connection Data (SCD)

Linkage information for the *records* of a *set occurrence*.

set occurrence

Single instance of a *set*. Comprises exactly one *owner record* and any number of subordinate *member records*.

set reference number

Sets are numbered contiguously in ascending order, beginning at 1.

set SEARCH KEY table

SEARCH KEY table for selecting a *member record* from a *set occurrence*.

SF pubset

See *single feature pubset*

shadow database

Backup of all the files of a database, each saved under the name ”*dbname*.*dbfile*.*copyname*”. A shadow database can be created at any time and processed parallel to the original database in RETRIEVAL mode. In addition BMEND can be used to apply *ALOG files* that have already been closed to the database parallel to the UDS/SQL *session*.

Shared user buffer pool

Shared buffer of several databases which is used in addition to the *System Buffer Pool*, solely for buffering *pages* of the *databases* that have been assigned to it.

SIA

See *Schema Information Area.*

SIB

See *SQL Interface Block.*

single feature pubset

A single feature pubset (SF pubset) consists of one or more homogeneous disks which must have the same major properties (disk format, allocation unit).

SLF

See *session log file*.

SM pubset

See *system managed pubset*

snap pair, snap pubset, snap session, snap unit

A snap unit is the copy of an (original) unit (logical disk in BS2000) at a particular time (“Point-in-Time copy”). The TimeFinder/Snap component creates this copy as a “snapshot” in accordance with the “Copy-On-First-Write strategy“: Only if data is modified is the original data concerned written beforehand into a central save pool of the Symmetrix system. The snap unit contains the references (track pointers) to the original data. In the case of unmodified data the references point to the unit, in the case of modified data to the save pool.
After they have been activated, the unit and snap unit are split; applications can access both.

The unit and snap unit together form a snap pair. TimeFinder/Snap manages this pair in what is known as a snap session.If snap units exist for all units of a pubset, these snap units together form the snap pubset.

Details of this are provided in the manual "Introduction to System Administration".

sort key table

Table pointing to the *member records* of a set *occurrence*.

source program

Program written in a programming language and not yet translated into machine language.

spanned record

Record exceeding the length of a *page*. **Only UDS/SQL-internal records** can be spanned records;
User record types must not exceed

- 2020 bytes for a page length of 2048 bytes
- 3968 bytes for a page length of 4000 bytes
- 8064 bytes for a page length of 8096 bytes.

SQL

SQL is a relational database language which has been standardized by ISO (International Organization for Standardization).

SQL conversation

See *conversation*.

SQL DML

SQL Data Manipulation Language for querying and updating data.

SQL Interface Block (SIB)

Interface between UDS/SQL and SQL application program(s); contains the SQL statement, any existing parameters and the statement results.

SQL transaction

Related sequence of *SQL* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.

SSIA

See *Subschema Information Area.*

SSIA-RECORD

UDS/SQL-internal *record type*, located in the *DBDIR*. *Records* belonging to this type are, for example, the Schema Information Area (*SIA*) and the Subschema Information Areas (*SSIAs*).

SSITAB module

Module generated by the BCALLSI utility routine; makes available the subschema information required by *CALL DML* programs.

SSL

See *Storage Structure Language.*

standard set

A *set* other than a *dynamic*, *implicit* or *SYSTEM set*.

statement code

Number stored in the first part of the *DATABASE-STATUS* item. Its function is to indicate which *DML* statement resulted in an exception condition.

status code

Number stored in the second part of the *DATABASE-STATUS* item. It indicates which exception condition has occurred.

Storage Structure Language (SSL)

Formalized language for describing the storage structure.

string

A series of consecutive alphanumeric characters.

subcontrol system

Component for the *independent DBH* . Responsible for control functions.

subschema

Section of a *schema* required for a particular *application* ; it can be restructured, within limits, for the intended application; a subschema is defined by means of the *Subschema DDL* .

Subschema DDL

Formalized language for defining a *subschema* .

Subschema Information Area (SSIA)

The SSIA contains all subschema information required by the *DBH* to carry out, on behalf of the user, the *database* accesses permitted within the specified *subschema* . The *DBH* loads the SSIA into main memory when it is referenced in a *READY* command.

subschema module

Module resulting from *subschema* compilation when a *COBOL DML* program is compiled. It must be linked in to the *application program* and includes the *USER WORK AREA* (*UWA*) as well as the *RECORD AREA* , which is also part of the *base interface block* (*BIB*). The name of the subschema module is the first 8 bytes of the subschema name.

subschema record

Record defined in the *Subschema DDL* .

SUB-SCHEMA SECTION

In COBOL programs with *DML* statements: section of the DATA DIVISION used for specifying the schema name and the subschema name.

subtransaction

In a distributed *transaction* , all the *processing chains* that address the databases in **one** *configuration* form a subtransaction.

system area

Realm required only by UDS/SQL. The system areas of a database include:

- the *Database Directory* (*DBDIR*),
- the *Database Compiler Realm* (*DBCOM*),
- the *COBOL Subschema Directory* (*COSSD*)

system break information

Indicates whether the *database* is consistent or inconsistent.

system buffer pools

Input/output buffer for database pages (see *page*). The buffer is part of the *common pool* (*independent DBH*) or the *DBH* work area (*linked-in DBH*). Its size is determined by the *DBH load parameters* 2KB-BUFFER-SIZE, 4KB-BUFFER-SIZE or 8KB-BUFFER-SIZE.

system managed pubset

A system managed pubset consists of one or more volume sets which, as with an *SF pubset*, comprise a collection of multiple homogeneous disks; here, too, homogeneity relates to particular physical properties such as disk format and allocation unit.

SYSTEM record

See *anchor record*.

SYSTEM set

Set whose *owner record type* is the symbolic *record type* SYSTEM.

T

table, multi-level

See *multi-level table*.

table (SQL)

A table in the context of *SQL* corresponds to a UDS/SQL *record type*.

table header

Contains general information on a table or *table page*:

- the table type and the level number of the table page,
- the number of reserved and current entries in this table page,
- the chaining reference to other table pages on the same level,
- the pointer to the associated table page on the next higher level,
- the pointer to the page containing the last table on the main level (for the highest-level table only).

table page

Page containing a table or part of a table. If a *table* which does not extend over several pages or the highest level of a multi-level *table* is concerned, "table page" only refers to the object involved, not the entire *page*.

TANGRAM

(Task and Group Affinity Management) Subsystem of BS2000 that plans the allocation of processors for task groups which access large quantities of shared data in multi-task applications.

task attribute TP

There are 4 task attributes in BS2000: SYS, TP, DIALOG and BATCH. Special runtime parameters that are significant for task scheduling are assigned to each of these task attributes. In contrast to the other task attributes, the TP attribute is characterized by optimized main memory management that is specially tailored to transaction processing requirements.

task communication

Communication between the *DBH modules* . See also *common pool* .

task deadlock

See *deadlock* .

task priority

In BS2000, it is possible to define a priority for a task. This priority is taken into account when initiating and activating the task.

Priorities may be fixed or variable. Variable priorities are adapted dynamically; fixed priorities do not change.

Note that UDS/SQL server tasks should be started with a fixed priority in order to ensure consistent performance.

TCUA

See *Transaction Currency Area.*

time acknowledgment

Message sent by the *UDS-D task* to the remote *application program* to indicate that there is still a *DML* statement being processed.

transaction (TA)

Related sequence of *DML* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.
For UDS-D:
The total set of *subtransactions* active at a given time.

transaction, committing a

Terminating a *transaction* with FINISH, i.e. all updates performed within the transaction are committed to the *database* .

transaction, rolling back a

Terminating a *transaction* with FINISH WITH CANCEL, i.e. all updates performed on the *database* within the transaction are rolled back.

Transaction Currency Area (TCUA)

Contains currency information.

transaction identification (TA-ID)

Assigned by the *DBH* to identify a particular *transaction*. Can be requested with the *DAL* command DISPLAY.

transfer pool

UDS-D-specific storage area in which the *UDSCT* receives the *BIBs* from *remote application programs*.

two-phase commit protocol

Procedure by which a *distributed transaction* that has made changes in at least one *remote configuration* is terminated in such a way as to safeguard *inter-configuration consistency* or UDS /SQL openUTM-D consistency. The two-phase commit is controlled

- by the distribution component in the *user task* if the *transaction* is distributed via UDS-D.
- by openUTM-D if the transaction is distributed via openUTM-D or via openUTM-D and UDS-D.

U

UDSADM

Module of the *independent DBH*; executes in the *administrator task.*

UDSHASH

Module generated by the BGSIA utility routine. It contains the names of all the *hash routines* defined in the *Schema DDL*.

UDSNET

Distribution component in the *user task*.

UDSSQL

Start module of the *independent DBH*; executes in the *master task*.

UDSSUB

Start module of the *independent DBH*; executes in the *server task*.

UDS-D task UDSCT

Task started for each *configuration* by UDS/SQL so that it can participate in distributed processing with UDS-D.

UDS/SQL / openUTM-D consistency

A *transaction* that has updated both openUTM data and UDS/SQL *databases* is terminated in such a way that the openUTM data and the UDS/SQL databases are either updated together or not at all.

UDS/SQL pubset declaration

Declaration in a *pubset declaration job variable* for restricting the UDS/SQL pubset environment. This reduces or prevents the risk of file names being ambiguous.

unique throughout the network

Unique in all the computers that are included in the *network*.

user database

The *realms* and files of the *database* required by the user in order to be able to store data in, and to retrieve data from a database are:

- the *Database Directory* (*DBDIR*),
- the *user realms*
- the module library for *hash routines* (*HASHLIB*).

user realm

A *realm* defined in the realm entry of the *Schema DDL*. It contains, among other things, the user records.

user task

Execution of an *application* program or openUTM program, including the parts linked by the system.

USER-WORK-AREA (UWA)

Transfer area for communication between the *application program* and the *DBH*.

UTM

See openUTM.

UWA

See *USER-WORK-AREA (UWA)* **.**

V

vector

Item with repetition factor. The repetition factor must be greater than 1. It specifies how many duplicates of the item are combined in the vector.

version number, internal

See *internal version number*.

W

warm start

A warm start is performed by UDS/SQL if an inconsistent *database* is attached to a *session*. For UDS/SQL this involves applying all updates of completed *transactions* to the database which have not yet been applied, *rolling back* all database transactions that are open, and making the database consistent. The related *RLOG file* and the *DB status file* are required for a warm start.

17 Abbreviations

ACS	Alias Catalog Service
Act-Key	ACTual KEY
AFIM	AFter-IMage
AP	Application Program
ASC	ASCending
BIB	Base Interface Block
BFIM	BeFore-IMage
COBOL	COmmon Business Oriented Language
CODASYL	COnference on DAta SYstem Languages
CRA	CuRrent of Area
CRR	CuRrent of Record
CRS	CuRrent of Set
CRU	Current of RunUnit
COSSD	COBOL SubSchema Directory
DAL	Database Administration Language
DB	DataBase
DBCOM	DataBase COmpiler Realm
DBDIR	DataBase DIRectory
DBH	DataBase Handler
DB-Key	DataBase Key
DBTT	DataBase key Translation Table
DDL	Data Description Language
DESC	DESCending
DML	Data Manipulation Language
DRV	Dual Recording by Volume
DSA	Database System Access
DSSM	Dynamic SubSystem Management
FC	Function Code
FPA	Free Place Administration
GS	Global Storage
HSMS	Hierarchic Storage Management System

ID	IDentification
IQL	Interactive Query Language
IQS	Interactive Query System
KDBS	Kompatible Datenbank-Schnittstelle (= compatible database interface)
KDCS	Kompatible Datenkommunikationsschnittstelle (= compatible data communications interface)
LM	Lock Manager
LMS	Library Maintenance System
MPVS	Multiple Public Volume Set
MR-NR	MainRef NumbeR
MT	Master Task
OLTP	OnLine Transaction Processing
openUTM	Universal Transaction Monitor
OT	Operator Task
PETA	Preliminary End of TrAnsaction
PPP	Probable Position Pointer
PTC	Prepared To Commit
PTT	Primäre Teiltransaktion (= primary subtransaction)
PVS	Public Volume Set
REC-REF	RECord REFerence number
RSQ	Record Sequence Number
SC	SubControl
SCD	Set Connection Data
SCI	Software Configuration Inventory
SECOLTP	SECure OnLine Transaction Processing
SECOS	SEcurity COntrol System
SET-REF	SET-REFerence
SIA	Schema Information Area
SIB	SQL Interface Block
SLF	Session Log File
SQL	Structured Query Language
SSD	Solid State Disk

SSIA	SubSchema Information Area
SSITAB	SubSchema Information TABle
SSL	Storage Structure Language
ST	ServerTask
STT	Sekundäre Teiltransaktion (= secondary subtransaction)
TA	TrAnsaction
TA-ID	TrAnsaction IDentification
TANGRAM	TAsk aNd GRoup Affinity Management
TCUA	Transaction CUrrency Area
UDS/SQL	Universal Database System/Structured Query Language
UWA	User Work Area

18 Related publications

You will find the manuals on the internet at https://bs2manuals.ts.fujitsu.com. You can order printed copies of those manuals which are displayed with an order number.

UDS/SQL (BS2000)
Application Programming
User Guide

UDS/SQL (BS2000)
Creation and Restructuring
User Guide

UDS/SQL (BS2000)
Design and Definition
User Guide

UDS/SQL (BS2000)
Messages
User Guide

UDS/SQL (BS2000)
Recovery, Information and Reorganization
User Guide

UDS/SQL (BS2000)
Ready Reference

UDS (BS2000)
Interactive Query System IQS
User’s Guide

UDS-KDBS (BS2000)
Compatible Database Interface
User Guide

SQL for UDS/SQL
Language Reference Manual

BS2000 OSD/BC
Commands
User Guide

BS2000 OSD/BC
Introduction to System Administration
User Guide

BS2000 OSD/BC
Executive Macros
User Guide

BS2000 OSD/BC
Introductory Guide to DMS
User Guide

SDF (BS2000)
SDF Dialog Interface
User Guide

SORT (BS2000)
User Guide

SPACEOPT (BS2000)
Disk Optimization and Reorganization
User Guide

LMS (BS2000)
SDF Format
User Guide

DSSM/SSCM
Subsystem Management in BS2000
User Guide

ARCHIVE (BS2000)
User Guide

DRV (BS2000)
Dual Recording by Volume
User Guide

HSMS / HSMS-SV (BS2000)
Hierarchical Storage Management System
Volume 1: Functions, Management and Installation
User Guide

SECOS (BS2000)
Security Control System
User Guide

openNet Server (BS2000)
BCAM
Reference Manual

DCAM (BS2000)
Program Interfaces
Reference Manual

DCAM (BS2000)
Macros
User Guide

OMNIS/OMNIS-MENU (BS2000)
Functions and Commands
User Guide

OMNIS/OMNIS-MENU (BS2000)
Administration and Programming
User Guide

openUTM
Concepts and Functions
User Guide

openUTM
Programming Applications with KDCS for COBOL, C and C++
User Guide

openUTM
Generating Applications
User Guide

openUTM
Administering Applications
User Guide

openUTM
Using openUTM Applications under BS2000
User Guide

openUTM
Messages, Debugging and Diagnostics in BS2000
User Guide

COBOL2000 (BS2000)
COBOL Compiler
Reference Manual

COBOL2000 (BS2000)
COBOL Compiler
User’s Guide

COBOL85 (BS2000)
COBOL Compiler
Reference Manual

COBOL85 (BS2000)
COBOL Compiler
User’s Guide

CRTE (BS2000)
Common Runtime Environment
User Guide

DRIVE/WINDOWS (BS2000)
Programming System
User Guide

DRIVE/WINDOWS (BS2000)
Programming Language
Reference Guide

DRIVE/WINDOWS (BS2000)
System Directory of DRIVE Statements
Reference Manual

DRIVE/WINDOWS (BS2000/SINIX)
Directory of DRIVE SQL Statements for UDS
Reference Manual

DAB (BS2000)
Disk Access Buffer
User Guide

Unicode in BS2000
Introduction

XHCS (BS2000)
8-Bit Code and Unicode Processing in BS2000
User Guide

BS2000 OSD/BC
Softbooks English
DVD

openSM2 (BS2000)
Software Monitor
User Guide

SNMP Management (BS2000)
User Guide