

# DB2 9 for z/OS: Distributed Functions

Establish connectivity to and from DB2 systems

Balance transaction workload across data sharing members

Explore the functions of Data Server Drivers and Clients



Paolo Bruni  
Nisanti Mohanraj  
Cristian Molaro  
Yasuhiro Ohmori  
Mark Rader  
Rajesh Ramachandran

**Redbooks**





International Technical Support Organization

**DB2 9 for z/OS: Distributed Functions**

July 2009

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xxiii.

### **First Edition (July 2009)**

This edition applies to Version 9.1 of IBM DB2 for z/OS (program number 5635-DB2).

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Figures</b> .....	ix
<b>Examples</b> .....	xv
<b>Tables</b> .....	xxi
<b>Notices</b> .....	xxiii
Trademarks .....	xxiv
<b>Preface</b> .....	xxv
The team that wrote this book .....	xxv
Become a published author .....	xxvii
Comments welcome .....	xxviii
<b>Summary of changes</b> .....	xxix
July 2009, First Edition .....	xxix
September 2009, First Update .....	xxix
May 2011, Second Update .....	xxix
<b>Part 1. Distributed database architecture and configurations</b> .....	1
<b>Chapter 1. Architecture of DB2 distributed systems</b> .....	3
1.1 Before you start .....	4
1.2 Distributed data access topology .....	4
1.2.1 Remote request .....	6
1.2.2 Remote unit of work .....	7
1.2.3 Distributed unit of work .....	8
1.2.4 Distributed request .....	9
1.3 DRDA .....	9
1.3.1 Functions and protocols of DRDA .....	11
1.3.2 Conversations between the AR and AS .....	12
1.3.3 Building blocks of DRDA .....	12
1.3.4 The DRDA process model .....	12
1.4 Implementation of DRDA in the DB2 family .....	14
1.4.1 DB2 for z/OS .....	14
1.4.2 DB2 for i .....	14
1.4.3 DB2 Server for VSE and VM .....	14
1.4.4 IBM Informix Dynamic Server .....	14
1.4.5 DB2 for Linux, UNIX and Windows .....	14
1.4.6 IBM Data Server Driver for ODBC and CLI .....	15
1.4.7 IBM Data Server Driver for JDBC and SQLJ .....	15
1.4.8 IBM Data Server Driver Package .....	16
1.4.9 pureQuery .....	16
1.5 Implementation of DRDA by non-IBM products .....	16
1.6 DB2 for z/OS Distributed Data Facility architecture .....	17
1.6.1 What DDF is .....	17
1.6.2 Distributed configurations .....	17
1.6.3 DDF implementation .....	18
1.6.4 Network protocols used by DDF .....	20
1.7 Connection pooling .....	22
1.7.1 Inactive connection support in a DB2 for z/OS server .....	22

1.7.2	Connection pooling using the IBM Data Server Drivers	24
1.7.3	Transaction pooling	25
1.8	Federated data support	26
1.8.1	IBM InfoSphere Federation Server	28
1.8.2	IBM InfoSphere Classic Federation Server for z/OS	29
<b>Chapter 2.</b>	<b>Distributed database configurations</b>	<b>33</b>
2.1	DB2 for z/OS both as a requester and a server	35
2.1.1	Basic configuration	35
2.1.2	Parallel sysplex environment	36
2.2	DB2 for LUW and DB2 for z/OS	37
2.2.1	DB2 for LUW ESE as requester to DB2 for z/OS server	37
2.2.2	DB2 for z/OS as requester to DB2 for LUW as server	37
2.2.3	DB2 for z/OS as an intermediate server	38
2.2.4	DB2 for z/OS as requester to a federation server	39
2.3	IBM Data Server Drivers and Clients as requesters	40
2.3.1	DB2 distributed clients: Historical view	41
2.3.2	IBM Data Server Drivers and Clients overview	41
2.3.3	Connecting to a DB2 data sharing group	46
2.3.4	Choosing the right configuration	47
2.3.5	Ordering the IBM Data Server Drivers and Clients	50
2.4	DB2 Connect to DB2 for z/OS: Past, present, and future	51
2.4.1	DB2 Connect Client as requester	52
2.4.2	DB2 Connect Server	53
2.4.3	Connecting to a DB2 data sharing group from DB2 Connect	55
2.4.4	DB2 Connect: A case of managing access to DB2 threads	56
2.5	DB2 Connect Server on Linux on IBM System z	58
2.5.1	DB2 Connect on Linux on z with HiperSockets	58
2.5.2	HiperSockets and DB2 data sharing configurations	61
2.6	DB2 for z/OS requester: Any (DB2) DRDA server	63
2.7	XA Support in DB2 for z/OS	63
<b>Part 2.</b>	<b>Setup and configuration</b>	<b>67</b>
<b>Chapter 3.</b>	<b>Installation and configuration</b>	<b>69</b>
3.1	TCP/IP setup	70
3.1.1	UNIX System Services setup	70
3.1.2	Language Environment considerations	71
3.1.3	Basic TCP/IP setup	71
3.1.4	TCP/IP settings in a data sharing environment	76
3.1.5	Sample DB2 data sharing DVIPA and Sysplex Distributor setup	79
3.1.6	Starting DDF with TCP/IP	83
3.2	DB2 system configuration	85
3.2.1	Defining the shared memory object	85
3.2.2	Configuring the Communications Database	86
3.2.3	DB2 installation parameters (DSNZPARM)	90
3.2.4	Updating the BSDS	97
3.2.5	DDF address space setup	103
3.2.6	Stored procedures and support for JDBC and SQLJ	104
3.3	Workload Manager setup	105
3.3.1	Enclaves	105
3.3.2	Managing DDF work with WLM	105
3.4	DB2 for LUW to DB2 for z/OS setup	114
3.4.1	IBM Data Server Drivers and Clients	114

3.4.2 DB2 Connect. . . . .	116
3.5 DRDA sample setup—From DB2 for z/OS requester to DB2 for LUW on AIX server . . . . .	122
3.6 Character conversion: Unicode. . . . .	125
3.7 Restrictions on the use of local datetime formats . . . . .	126
3.8 HiperSockets: Definition . . . . .	127
<b>Chapter 4. Security . . . . .</b>	<b>129</b>
4.1 Guidelines for basic DRDA security setup over TCP/IP . . . . .	130
4.1.1 Security options supported by DRDA access to DB2 for z/OS. . . . .	130
4.1.2 Authorization . . . . .	132
4.1.3 Important considerations when setting security related DSNZPARMs. . . . .	132
4.1.4 Recommendation for tightest security. . . . .	135
4.2 DRDA security requirements for an application server . . . . .	140
4.2.1 Characteristics of a typical application server security model. . . . .	140
4.2.2 Considerations for DRDA security behind the application server . . . . .	141
4.2.3 Identifying a client user coming from the application server . . . . .	142
4.2.4 Network trusted context and roles. . . . .	145
4.3 Encryption options. . . . .	147
4.3.1 DRDA encryption . . . . .	147
4.3.2 IP Security. . . . .	150
4.3.3 Secure Socket Layer. . . . .	151
4.3.4 DataPower . . . . .	172
4.4 Addressing dynamic SQL security concerns. . . . .	173
4.4.1 Using DYNAMICRULES(BIND) to avoid granting table privileges . . . . .	173
4.4.2 Using stored procedures for static SQL security benefits. . . . .	175
4.4.3 Static SQL options of JDBC to realize static SQL security benefits . . . . .	176
4.4.4 Static execution of dynamic SQL to benefit from static SQL security . . . . .	176
<b>Part 3. Distributed applications . . . . .</b>	<b>183</b>
<b>Chapter 5. Application programming . . . . .</b>	<b>185</b>
5.1 Accessing data on a DB2 for z/OS server from a DB2 for z/OS requester. . . . .	186
5.1.1 System-directed versus application-directed access . . . . .	186
5.1.2 Program preparation when DB2 for z/OS is the AR . . . . .	190
5.1.3 Using DB2 for z/OS as a requester going outbound to a non-DB2 for z/OS server. . . . .	193
5.2 Migrating from DB2 private protocol to DRDA. . . . .	194
5.2.1 DB2 performance trace to show private protocol use . . . . .	195
5.2.2 The PRIVATE to DRDA REXX migration tool: DSNTDP2DP . . . . .	195
5.3 Program preparation steps when using non-DB2 for z/OS Requesters . . . . .	200
5.3.1 Connecting and binding packages from DB2 CLP . . . . .	200
5.3.2 Using the DB2Binder utility to bind packages used by the Data Server Drivers. . . . .	203
5.4 Using the non-Java-based IBM Data Server Drivers . . . . .	203
5.4.1 Using the IBM Data Server Driver for ODBC and CLI. . . . .	203
5.4.2 Using the IBM Data Server Driver Package in a .NET environment. . . . .	204
5.4.3 db2cli.ini and db2dsdriver.cfg . . . . .	206
5.5 Using the IBM Data Server Driver for JDBC and SQLJ . . . . .	207
5.5.1 Connecting to a DB2 for z/OS server using the Type 4 driver . . . . .	208
5.5.2 Coding static applications using SQLJ . . . . .	209
5.6 Developing static applications using pureQuery . . . . .	211
5.6.1 When should you use pureQuery? . . . . .	212
5.6.2 pureQuery programming styles. . . . .	213

5.6.3 pureQuery client optimization . . . . .	213
5.7 Remote application development . . . . .	216
5.7.1 Limited block fetch . . . . .	216
5.7.2 Multi-row FETCH . . . . .	217
5.7.3 Understanding the differences between limited block FETCH and multi-row FETCH . . . . .	219
5.7.4 Fast implicit CLOSE and COMMIT of cursors . . . . .	219
5.7.5 Multi-row INSERT . . . . .	220
5.7.6 Multi-row MERGE . . . . .	221
5.7.7 Heterogeneous batch updates . . . . .	221
5.7.8 Progressive streaming . . . . .	222
5.7.9 SQL Interrupts . . . . .	223
5.7.10 Remote external stored procedures and native SQL procedures . . . . .	224
5.8 XA transactions . . . . .	225
5.8.1 Using the Type 4 driver to enable direct XA transactions . . . . .	226
5.8.2 Using the IBM non-Java-based Data Server Drivers/Clients to enable direct XA transactions . . . . .	227
5.9 Remote application recommendations . . . . .	229
<b>Chapter 6. Data sharing . . . . .</b>	<b>233</b>
6.1 High availability aspects of DRDA access to DB2 for z/OS . . . . .	234
6.1.1 Key components for DRDA high availability . . . . .	234
6.1.2 z/OS WLM in DRDA workload balancing . . . . .	234
6.1.3 The sysplex awareness of clients and drivers . . . . .	239
6.1.4 Network resilience using Virtual IP Addressing . . . . .	242
6.1.5 Advanced high availability for DB2 for z/OS data sharing . . . . .	243
6.1.6 Scenario with Q-Replication for high availability . . . . .	246
6.2 Recommendations for common deployment scenarios . . . . .	247
6.2.1 DB2 data sharing subsetting . . . . .	248
6.2.2 Application Servers . . . . .	249
6.2.3 Distributed three-tier DRDA clients . . . . .	256
6.3 DB2 failover scenario with and without Sysplex Distributor . . . . .	259
6.3.1 Configuration for scenarios . . . . .	259
6.3.2 Application states for scenarios . . . . .	260
6.3.3 Results without Sysplex Distributor . . . . .	261
6.3.4 Results with Sysplex Distributor . . . . .	264
6.4 Migration and coexistence . . . . .	265
<b>Part 4. Performance and problem determination . . . . .</b>	<b>267</b>
<b>Chapter 7. Performance analysis . . . . .</b>	<b>269</b>
7.1 Application flow in distributed environment . . . . .	270
7.2 System topics . . . . .	271
7.2.1 Database Access Threads . . . . .	271
7.2.2 Accumulation of DDF accounting records . . . . .	274
7.2.3 zIIP . . . . .	279
7.2.4 Using RMF to monitor distributed data . . . . .	288
7.3 Checking settings in a distributed environment . . . . .	294
7.3.1 db2set: DB2 profile registry command . . . . .	294
7.3.2 db2 get dbm cfg . . . . .	295
7.3.3 db2 get cli configuration . . . . .	296
7.3.4 db2pd . . . . .	297
7.3.5 Getting database connection information . . . . .	298
7.3.6 Getting online help for db2 commands . . . . .	301

7.3.7 Other useful sources of information . . . . .	303
7.4 Obtaining information about the host configuration. . . . .	307
7.4.1 Verification of currently active DSNZPARMs . . . . .	307
7.4.2 SYSPLAN and SYSPACKAGES . . . . .	309
7.4.3 Resource Limit Facility . . . . .	310
7.4.4 GET_CONFIG and GET_SYSTEM_INFO stored procedures . . . . .	312
7.4.5 DB2 commands . . . . .	315
<b>Chapter 8. Problem determination.</b> . . . .	323
8.1 Traces at DB2 Client and DB2 Gateway. . . . .	324
8.1.1 The n-tier message communication . . . . .	324
8.1.2 CLI traces . . . . .	325
8.1.3 DRDA traces . . . . .	333
8.1.4 JDBC traces . . . . .	338
8.2 Accounting for distributed data with the EXCSQLSET command. . . . .	343
8.2.1 TCP/IP packet tracing on z/OS . . . . .	350
8.3 Network analyzer . . . . .	355
8.4 DB2 for z/OS tracing capabilities . . . . .	363
8.4.1 Collecting traces . . . . .	364
8.4.2 Using OMEGAMON PE for reporting . . . . .	372
8.4.3 IFCIDs for DRDA problem determination . . . . .	377
8.4.4 WLM classification rules and accounting information . . . . .	387
8.4.5 Step-by-step performance analysis. . . . .	390
<b>Part 5. Appendixes</b> . . . . .	393
<b>Appendix A. DRDA-related maintenance</b> . . . . .	395
A.1 Recent DRDA APARs. . . . .	396
<b>Appendix B. Configurations and workload.</b> . . . .	401
B.1 Configurations. . . . .	402
B.2 The TRADE workload. . . . .	404
B.2.1 IBM Trade performance benchmark sample for WebSphere Application Server V6.1. . . . .	404
B.2.2 TRADE installation . . . . .	405
B.3 Using Trade . . . . .	416
<b>Appendix C. Sample applications</b> . . . . .	419
C.1 Sample Java program to call a remote native SQL procedure. . . . .	420
C.1.1 Using the Type 4 driver to call a native SQL procedure (BSQLAlone.Java) . . . . .	420
C.2 XA transaction samples . . . . .	423
C.2.1 createRegisterXADS.java. . . . .	423
C.2.2 Test application for XA transaction (XATest.Java). . . . .	426
C.3 Progressive streaming . . . . .	439
C.3.1 Progressive streaming: XMLTest_RedJava . . . . .	439
<b>Appendix D. Sample programs for performance analysis</b> . . . . .	443
D.1 Stress tests script . . . . .	444
D.2 REXX parser of GTF trace for IFCID 180. . . . .	446
<b>Abbreviations and acronyms</b> . . . . .	457
<b>Related publications</b> . . . . .	461
IBM Redbooks . . . . .	461
Other publications . . . . .	461

Online resources .....	462
How to get Redbooks .....	463
Help from IBM .....	463
<b>Index</b> .....	465

# Figures

1-1	Unit of work concept	5
1-2	Two-phase commit protocol	5
1-3	Remote request	7
1-4	Remote unit of work	8
1-5	Distributed unit of work	9
1-6	Distributed request	9
1-7	DRDA network	10
1-8	Functions and protocols used by DRDA	11
1-9	DRDA process model	13
1-10	Connecting to DDF	18
1-11	Address spaces	19
1-12	DDF - Using shared private storage	20
1-13	Communication using TCP/IP	21
1-14	Communication using SNA	22
1-15	Inactive connection support	23
1-16	Connection pooling	24
1-17	Transaction pooling	25
1-18	Sysplex workload balancing	26
1-19	InfoSphere Federation Server products	27
1-20	InfoSphere Federation Server	29
1-21	Classic Federation Server for z/OS	29
1-22	Supported data sources	30
2-1	Connecting from DB2 for z/OS to DB2 for z/OS	35
2-2	Connecting to a DB2 data sharing group from a DB2 for z/OS system	36
2-3	DB2 for LUW ESE requester connecting to DB2 for z/OS	37
2-4	DB2 for z/OS requester connecting to DB2 for LUW ESE or DB2 for LUW WSE	38
2-5	DB2A as an intermediate server between a requester and a server	38
2-6	DB2 for z/OS as requester in a federation server, unprotected update scenario	39
2-7	IBM Data Server Driver for JDBC and SQLJ connecting directly to DB2 for z/OS	42
2-8	IBM Data Server Driver for ODBC and CLI connecting directly to DB2 for z/OS	43
2-9	IBM Data Server Driver Package connecting directly to DB2 for z/OS	44
2-10	IBM Data Server Runtime Client connecting directly to DB2 for z/OS	44
2-11	IBM Data Server Client connecting directly to DB2 for z/OS	45
2-12	IBM Data Server Drivers connecting to a DB2 data sharing group	46
2-13	Current configuration with DB2 Connect Client and Server	49
2-14	DB2 Connect Client and Server replaced with IBM Data Server Drivers	50
2-15	DB2 Connect Client connecting to DB2 for z/OS	52
2-16	DB2 Connect Server providing access to DB2 for z/OS	53
2-17	Example of DB2 Connect Server providing connection concentration	54
2-18	DB2 Connect Server in a Web application server environment	54
2-19	DB2 Connect Client connecting to a DB2 data sharing group	55
2-20	DB2 Connect Server providing access to a DB2 data sharing group	56
2-21	Controlling DB2 threads with a DB2 Connect Server	58
2-22	HiperSocket: Example of multiple LPAR communication	59
2-23	Server consolidation and HiperSockets and Linux on System z for DB2 Connect	60
2-24	HiperSocket: DB2 Connect using HiperSocket to communicate with DB2 for z/OS	61
2-25	HiperSockets in a data sharing environment	62
2-26	XA transaction support with DB2 Connect or WebSphere Application Server	64

2-27	XA transaction support without DB2 Connect . . . . .	64
2-28	XA transaction support in a DB2 data sharing environment . . . . .	65
3-1	Output of D9C1DIST started task as seen from SDSF . . . . .	70
3-2	Output of LISTUSER STC OMVS command . . . . .	70
3-3	JCL of TCP/IP job, from SDSF display, showing high level qualifier for TCP/IP . . . . .	74
3-4	Starting point for TCP.HOSTS.LOCAL . . . . .	76
3-5	VipaDynamic statements including Sysplex Distributor definition for SC70 . . . . .	78
3-6	VipaDynamic statements with backup SD for SC64 . . . . .	78
3-7	VipaDynamic statements with backup SD for SC63 . . . . .	78
3-8	BSDS specification with member-specific DVIPA and group DVIPA . . . . .	79
3-9	DB2 for z/OS V8: Port statements binding a specific IP address to the DB2 ports. . . . .	79
3-10	Output of D TCPIP,,N,CONN command . . . . .	80
3-11	Contents of /SC63/etc/hosts including DVIPA addresses . . . . .	81
3-12	Output of D TCPIP,,NETSTAT,HOME command . . . . .	82
3-13	Output of D TCPIP,,N,CONN command . . . . .	83
3-14	Example of DISPLAY DDF from DB9A standalone DB2 . . . . .	84
3-15	Example of DISPLAY DDF from D9C1 data sharing member . . . . .	84
3-16	Example of DISPLAY DDF from D9C2 data sharing member . . . . .	84
3-17	Example of DISPLAY DDF from D9C3 data sharing member . . . . .	85
3-18	Results of DISPLAY VIRTSTOR,HVSHARE showing default definition . . . . .	86
3-19	DSNTIPE panel specifying MAXDBAT and CONDBAT values . . . . .	91
3-20	DSNTIPR panel showing DDF values for subsystem DB9A . . . . .	93
3-21	DSNTIP5 panel showing values for subsystem DB9A . . . . .	95
3-22	DSNJU003 for DB9A BSDS . . . . .	98
3-23	DB9A DSNJU004 output with DDF values . . . . .	98
3-24	DSNJU003 for D9C1 BSDS . . . . .	99
3-25	D9C1 DSNJU004 output for member D9C1 . . . . .	99
3-26	DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C1 . . . . .	100
3-27	DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C2 . . . . .	100
3-28	DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C3 . . . . .	100
3-29	DSNJU004 output for member D9C1 with DVIPA specified . . . . .	101
3-30	Output from -D9C1 DISPLAY DDF showing DVIPA specifications . . . . .	101
3-31	Output from -D9C1 DISPLAY DDF showing DNS support for the group . . . . .	102
3-32	DSNJU003 input to add ALIAS definitions to D9C1 . . . . .	102
3-33	DSNJU003 input to add ALIAS definition to D9C2 . . . . .	103
3-34	DSNJU004 output showing LOCATION ALIAS and DVIPA with IPV4 . . . . .	103
3-35	JCL for D9C1DIST . . . . .	104
3-36	WLM: Choosing Classification Rules . . . . .	107
3-37	WLM: Subsystem Type Selection: Choosing classification rules for started tasks . . . . .	107
3-38	WLM: STC service classes and report classes . . . . .	108
3-39	WLM: Transaction Name Group (TNG) for all DB2s in our sysplex . . . . .	108
3-40	WLM: STCHI service class goal . . . . .	109
3-41	SDSF display showing service classes for D9C1 address spaces . . . . .	109
3-42	WLM: DDFONL service class goals . . . . .	110
3-43	WLM: DDFDEF service class goals . . . . .	110
3-44	WLM: DDFBAT service class goals for our default DDF service class . . . . .	111
3-45	WLM: DDFTOT service class definition . . . . .	111
3-46	WLM: DDFTST service class definition . . . . .	112
3-47	A subset of WLM classification rules . . . . .	113
3-48	Sample db2dsdriver.cfg for our environment . . . . .	115
3-49	Sample db2dsdriver.cfg provided with the driver . . . . .	116
3-50	Two-tier configuration to our standalone DB2 for z/OS, DB9A . . . . .	117
3-51	Three-tier configuration to our standalone DB2 for z/OS . . . . .	118



3-52	Two-tier connection to our DB2 for z/OS data sharing group . . . . .	120
3-53	Three-tier connection to our DB2 for z/OS data sharing group. . . . .	121
3-54	Steps to configure DB2 for z/OS as DRDA AR to DB2 for LUW as DRDA AS. . . . .	123
3-55	DSNTIPF panel where you specify your system CCSIDs. . . . .	126
3-56	TCP profile extract with HiperSocket definitions - part 1. . . . .	127
3-57	TCP profile extract with HiperSocket definitions - part 2. . . . .	128
4-1	DRDA connection flow of DB2 for z/OS . . . . .	132
4-2	Connecting to DB2 using RACF PassTickets. . . . .	136
4-3	Authentication process using Kerberos . . . . .	138
4-4	Setting for the DataSource custom properties on WebSphere Application Server. . . . .	139
4-5	Syntax diagram for the DSNLEUSR . . . . .	139
4-6	Typical DRDA access Web application server security model . . . . .	141
4-7	Client Informations settings through WebSphere Application Server admin console . . . . .	142
4-8	Client information setting example using datasource ODBC settings. . . . .	143
4-9	Configure WebSphere Application Server DataSource custom properties to use AES encryption. . . . .	149
4-10	IPSec overview . . . . .	150
4-11	The padlock symbol indicates encryption . . . . .	151
4-12	SSL overview . . . . .	152
0-1	BSDS . . . . .	153
4-13	Define RACF resources for policy agent. . . . .	154
4-14	Definition for policy agent started task . . . . .	155
4-15	Definition for policy agent environment file . . . . .	155
4-16	Definition for policy agent main configuration file . . . . .	155
4-17	Sample server definition for AT-TLS configuration file . . . . .	156
4-18	Add TTLS parameter to TCP/IP stack configuration. . . . .	157
4-19	Activate DIGTCERT and DIGTRING class . . . . .	157
4-20	Create a self-signed server CA certificate . . . . .	157
4-21	Create private server certificate . . . . .	158
4-22	Create server keyring and add certificate . . . . .	158
4-23	Export server CA certificate . . . . .	159
4-24	Change BSDS to enable the secured port . . . . .	159
4-25	The IBM Key Management tool window . . . . .	162
4-26	Create a new Key Database file . . . . .	163
4-27	Setting password for key database and making stash file . . . . .	164
4-28	Import the DB2 server certificate to Key Database . . . . .	164
4-29	Enter the label for the DB2 server Certificate . . . . .	165
4-30	The key database after imported the DB2 server certificate. . . . .	165
4-31	Settings for \$DB2ISNTPORF and \$DB2INSTDEF DB2 profile variables. . . . .	166
4-32	The network capture of DRDA request (using Wireshark) . . . . .	170
4-33	The network capture of DRDA Data encrypt. . . . .	171
4-34	The network capture of DRDA with SSL enabled. . . . .	171
4-35	Sample datasource custom properties settings for WebSphere Application Server. . . . .	172
4-36	Overview of preparation steps to execute JDBC application in static mode. . . . .	177
5-1	Connection management . . . . .	187
5-2	Execution of remote packages . . . . .	193
5-3	Options for tool DSNTP2DP . . . . .	196
5-4	Sample output for packages . . . . .	197
5-5	Sample output from PLANS data set . . . . .	199
5-6	.NET application code sample . . . . .	205
5-7	.NET configuration file. . . . .	206
5-8	SQLJ application sample . . . . .	210
5-9	Customizing and binding an SQLJ application . . . . .	211

5-10 pureQuery runtime . . . . .	212
5-11 The internal workings of pureQuery . . . . .	215
5-12 Enabling XA transaction . . . . .	228
5-13 .NET application that uses XA transactions . . . . .	229
6-1 DRDA access to DB2 data sharing . . . . .	240
6-2 The logical flow of DRDA AR clients obtaining server information . . . . .	241
6-3 Sysplex Distributor connection assignment . . . . .	244
6-4 Client reroute . . . . .	246
6-5 Multi-sysplex configuration scenario . . . . .	247
6-6 DB2 data sharing subsetting . . . . .	248
6-7 Java application server scenario configuration using WebSphere Application Server . . . . .	250
6-8 Setting Type 4 driver configuration properties file to WebSphere Application Server . . . . .	252
6-9 Non-Java-based application server scenario configuration using .NET . . . . .	254
6-10 Distributed three tier DRDA clients scenario . . . . .	257
6-11 Failover test scenario with Sysplex Distributor . . . . .	260
7-1 2 tier architecture representation . . . . .	270
7-2 3-tier architecture representation . . . . .	270
7-3 Effects of ACCUMACC on some of the fields of the accounting records . . . . .	278
7-4 Reverting to ACCUMACC=NO . . . . .	279
7-5 Accounting report including zIIP CPU usage . . . . .	284
7-6 CPU report with zIIP and zAAP . . . . .	290
7-7 Calculating zIIP redirect % . . . . .	291
7-8 RMF Spreadsheet reporter: creating a working set menu . . . . .	292
7-9 RMF Spreadsheet reporter: creating a working set . . . . .	293
7-10 RMF Spreadsheet reporter: selecting reports . . . . .	293
7-11 RMF report example: Physical Total Dispatch Time % . . . . .	294
7-12 DB2 PE System parameters GUI view . . . . .	308
7-13 OSC Panel DSNZPARMs . . . . .	309
7-14 OMEGAMON PE showing the execution of a DISPLAY RLIMIT command . . . . .	311
7-15 START RLIMIT example . . . . .	311
8-1 3-tier architecture simplified . . . . .	324
8-2 2-tier architecture simplified . . . . .	324
8-3 Traces and tools in a n-tier environment . . . . .	325
8-4 CLI settings in the Configuration Assistant . . . . .	328
8-5 Add CLI parameters in the Configuration Assistant . . . . .	328
8-6 CLI Settings panel . . . . .	329
8-7 CLI/ODBC Setting, Windows Control panel . . . . .	329
8-8 Changing the CLI setting using the DB2 Configuration Assistant . . . . .	346
8-9 CLI settings: Transaction section . . . . .	346
8-10 Accounting information in OMEGAMON PE . . . . .	349
8-11 Network analyzer options panel . . . . .	356
8-12 Network analyzer main panel . . . . .	357
8-13 DRDA SECMEC DDM code point monitored by Wireshark . . . . .	358
8-14 Apply DRDA protocol filter to Wireshark capture . . . . .	359
8-15 Filtering packets in Wireshark . . . . .	359
8-16 Analysis of DRDA packets in a spreadsheet . . . . .	360
8-17 Configuration Assistant server authentication: No encryption . . . . .	361
8-18 Password and user ID in network analyzer . . . . .	362
8-19 Configuration assistant enable encryption option . . . . .	362
8-20 Configuration Assistant Server Authentication, Enable encryption option selected . . . . .	363
8-21 Creating spreadsheets reports from OMEGAMON Warehouse tables . . . . .	374
8-22 Custom DDF Statistics chart, blocks sent by statistics interval . . . . .	375
8-23 Custom DDF accounting chart, DBAT wait time versus commits per location . . . . .	376

8-24 Analysis of zIIP utilization by workstation name . . . . .	377
8-25 Mapping the GTF header . . . . .	384
8-26 WLM workload classification . . . . .	388
8-27 Flowchart describing the problem determination method in n-tier environment . . . .	390
B-1 Starting z/OS and AIX configurations . . . . .	402
B-2 Configuration after implementing DVIPA support, plus Linus on IBM System z. . . . .	403
B-3 Location subsets in our DB2 data sharing group. . . . .	404
B-4 Trade overview . . . . .	405
B-5 WebSphere Application Server admin console after installation . . . . .	413
B-6 JDBC Provider defined from configuration script . . . . .	413
B-7 TradeDataSource from WebSphere Application Server admin console . . . . .	414
B-8 Modify DataSource to Type 4 Connection . . . . .	414
B-9 Restart application server from WebSphere Application Server admin console . . . .	415
B-10 Finish installation by populating Trade Database . . . . .	415
B-11 Verify your installation by logging into Trade . . . . .	416
B-12 Trade home panel . . . . .	417
B-13 Test Trade scenario . . . . .	417

Archived

# Examples

2-1	Updating tables in order in a single UOW . . . . .	40
3-1	Displaying an OMVS user . . . . .	70
3-2	Defining a superuser . . . . .	71
3-3	Port reservations for two DB2 subsystems . . . . .	74
3-4	Port reservations for three members of a DB2 data sharing group . . . . .	77
3-5	Port reservation statements for our three-way data sharing group . . . . .	81
3-6	Port reservations for three members including aliases . . . . .	82
4-1	Authentication mechanism by catalog database command . . . . .	131
4-2	Authentication rejection with and without extended security . . . . .	134
4-3	Change RACF password on connect . . . . .	135
4-4	Activates PTKTDATA class . . . . .	136
4-5	New profiles to remote DB2 subsystem . . . . .	136
4-6	Catalog a database using Kerberos authentication . . . . .	137
4-7	Example for setting authentication mechanism at DB2 Connect . . . . .	138
4-8	Example of executing SYSPROC.DSNLEUSR(from DB2 Connect). . . . .	139
4-9	Display of inserted row of SYSPROC.DSNLEUSR. . . . .	140
4-10	Sample configuration file contents of data server driver . . . . .	143
4-11	Sample for setting client information from Java applications . . . . .	144
4-12	Sample setting client information from WebSphere Application Server applications . . . . .	144
4-13	Sample setting client information in ODBC/CLI applications . . . . .	145
4-14	Sample setting client information in ADO.NET application (Visual Basic) . . . . .	145
4-15	SYSLOG output from misconfiguration . . . . .	147
4-16	Catalog database with AES option . . . . .	148
4-17	Sample data server driver configuration for AES encryption . . . . .	148
4-18	Using AES from Java application . . . . .	148
4-19	Catalog database with DRDA data stream encryption . . . . .	149
4-20	Sample data server driver configuration for data stream encryption . . . . .	149
4-21	Using data stream encryption for Java applications . . . . .	150
4-22	Generating keystore for Java clients . . . . .	160
4-23	Import server certificate to your keystore . . . . .	161
4-24	List the keystore entry . . . . .	161
4-25	Starting the IBM Key Management tool . . . . .	162
4-26	Contents of SSLClientconfig.ini . . . . .	166
4-27	sample Java code for SSL connection . . . . .	167
4-28	Executing Java application using SSL . . . . .	168
4-29	Sample db2dsdriver.cfg configuration . . . . .	168
4-30	Catalog DB2 sever to use SSL connection to DB2 Connect . . . . .	169
4-31	Sample SSL connection using DB2 Connect . . . . .	169
4-32	Sample SYSLOG output for failed SSL connection . . . . .	170
4-33	The db2cli.ini configuration for static SQL profiling capture mode . . . . .	178
4-34	Sample CLI script used to test static profiling . . . . .	179
4-35	Static profiling capture file . . . . .	179
4-36	db2cli.ini configuration for static SQL profiling match mode . . . . .	180
4-37	Capture - match log file . . . . .	180
5-1	Three-part table names . . . . .	187
5-2	Aliases for three-part table names . . . . .	187
5-3	Incorrect alias definition . . . . .	188
5-4	Correct remote alias definition . . . . .	188

5-5	Explicit CONNECT statement . . . . .	188
5-6	Binding packages at a remote site . . . . .	190
5-7	Binding packages into the plan . . . . .	191
5-8	Binding remote SPUFI packages . . . . .	194
5-9	Connecting from DB2 CLP . . . . .	200
5-10	Binding DB2 Connect packages on a remote DB2 for z/OS . . . . .	200
5-11	Using command db2bfd . . . . .	201
5-12	Executing remote SQL interactively from DB2 CLP . . . . .	202
5-13	Creating and calling a native SQL procedure from DB2 CLP . . . . .	202
5-14	Using the DB2Binder utility . . . . .	203
5-15	Bind options as a property for the DB2Binder class . . . . .	203
5-16	Connecting to DB2 for z/OS through CLI . . . . .	203
5-17	The db2dsdriver.cfg file . . . . .	204
5-18	Determining the driver version . . . . .	208
5-19	Using the getConnection() . . . . .	208
5-20	Connecting to DB2 for z/OS through the DataSource interface . . . . .	208
5-21	Capturing dynamic SQL statements . . . . .	213
5-22	Configuring target packages . . . . .	213
5-23	Using the StaticBinder utility . . . . .	214
5-24	Running the static application . . . . .	214
5-25	Block fetching in a Java program . . . . .	216
5-26	Retrieving a rowset . . . . .	218
5-27	Implicit multi-row fetching in a Java program . . . . .	218
5-28	Setting a property to enable extended diagnostic . . . . .	219
5-29	Multi-row INSERT in a PL/I program . . . . .	220
5-30	Using addBatch() in a Java program . . . . .	221
5-31	Multi-row MERGE in a Java program . . . . .	221
5-32	Issuing SQL interrupts . . . . .	223
5-33	Creating and calling a native SQL procedure using the Type 4 driver . . . . .	224
5-34	Enabling XA transaction through explicit XA API . . . . .	226
6-1	RMF workload activity report example . . . . .	235
6-2	Example of the server list from the DISPLAY DDF DETAIL command . . . . .	236
6-3	Example trace output from the global transport objects pool . . . . .	238
6-4	Display server list information from DB2 Connect Server . . . . .	238
6-5	Verify level of Type 4 driver . . . . .	249
6-6	Example of configuration properties file . . . . .	252
6-7	Sample setting for db2dsdriver.cfg . . . . .	255
7-1	DIS THD(*) DETAIL . . . . .	272
7-2	OM/PE Statistics Report Short showing DBAT QUEUED-MAXIMUM ACTIVE > 0 . . . . .	272
7-3	OM/PE Statistics Report Long showing DBAT QUEUED-MAXIMUM ACTIVE > 0 . . . . .	273
7-4	Thread cancelled because of idle thread timeout threshold reached . . . . .	274
7-5	Activating accounting rollup . . . . .	277
7-6	SET SYSPARM RELOAD command example . . . . .	277
7-7	De-Activating accounting rollup . . . . .	279
7-8	Output of /d m=cpu command . . . . .	281
7-9	Extract of hlq.SDSNMACS(DSNDQWAC) macro showing zIIP related fields . . . . .	285
7-10	OMEGAMON PE RECTRACE command example . . . . .	286
7-11	OMEGAMON PE Record Trace extract showing zIIP related fields . . . . .	286
7-12	SDSF view of enclaves showing zIIP utilization . . . . .	288
7-13	RMF online monitoring of enclaves . . . . .	288
7-14	RMF Enclave Classification Data . . . . .	289
7-15	RMF batch reporting . . . . .	290
7-16	db2set -all . . . . .	295

7-17	<b>db2set -lr</b> command	295
7-18	<b>db2 get dbm cfg</b> output	295
7-19	Getting the port on which DB2 for LUW listen	296
7-20	GET CLI CONFIGURATION syntax	296
7-21	GET CLI CONFIGURATION output example	297
7-22	<b>db2pd</b> sysplex syntax	297
7-23	<b>db2pd</b> usage example: Getting the server list	298
7-24	Syntax of the list database directory command	298
7-25	<b>db2 list db</b> directory command output example	298
7-26	Syntax of the list node directory command	300
7-27	<b>list node directory</b> command output	300
7-28	Syntax of the list dcs directory command	300
7-29	<b>list dcs directory</b> command output example	301
7-30	Getting online help using the CLP	302
7-31	Using the -h option with a system command	302
7-32	Using the command <b>db21icm</b>	303
7-33	Using the command <b>db2level</b>	303
7-34	Getting the service associated with the DB2 Server	304
7-35	Getting the port number from /etc/services	304
7-36	Getting the IP address of a DB2 Connect or ESE server from its dns entry	304
7-37	Getting your IP address in a windows machine	304
7-38	Getting the IP address of an AIX server	305
7-39	DB2 SYSPRINT	305
7-40	<b>ping -a</b> example command for resolving a host name	305
7-41	Updating <b>db2 diaglevel</b>	306
7-42	Extract of and OMEGAMON PE System Parameter Report	307
7-43	STOP RLIMIT example	312
7-44	Example of Java program calling SYSPROC.GET_INFO	312
7-45	Execution a Java program calling SYSPROC.GET_INFO	314
7-46	Extract of <b>xml_output.xml</b> file	314
7-47	DIS THD output example	315
7-48	Display of INACTIVE THREADS	316
7-49	<b>DIS THD(*) LOCATION(::9.12.4.121)</b> command output	316
7-50	DISPLAY LOCATION example	317
7-51	DIS DDF syntax	317
7-52	DIS DDF command example	318
7-53	DIS DDF DETAIL example	318
7-54	DIS THD(*) TYPE(INACTIVE) command output	319
7-55	STOP DDF command syntax	320
7-56	STOP DDF output	320
7-57	DISPLAY DFF output (suspending)	321
7-58	DISPLAY LOCATION output	321
7-59	DISPLAY DDF after RESUME	321
8-1	GET CLI CONFIGURATION syntax	330
8-2	GET CLI CONFIGURATION output example	330
8-3	Execution of UPDATE CLI CFG commands	330
8-4	GET CLI CFG command output after changes	331
8-5	How to find the <b>db2cli.ini</b> file in an AIX server	331
8-6	Using the <b>db2set</b> command to display the variable <b>DB2CLIINIPATH</b>	332
8-7	COMMON section of <b>db2cli.ini</b>	332
8-8	CLI trace example	332
8-9	Locating the <b>db2drdat</b> in DB2 Connect and ESE in AIX	334
8-10	<b>db2drdat</b> command syntax	334

8-11	db2drdat on . . . . .	335
8-12	Stopping the db2drdat traces . . . . .	335
8-13	DRDA trace example . . . . .	335
8-14	Enabling traces using the DriverManager interface . . . . .	338
8-15	Sample modification to connection string . . . . .	338
8-16	Sample JCC Type 4 trace file (partial output) . . . . .	339
8-17	Sample SQLJ trace file (partial output) . . . . .	340
8-18	Sample JCC Type 4 trace file (partial output) showing SQL Error . . . . .	341
8-19	Starting db2trc. . . . .	341
8-20	<b>db2trc dump</b> command . . . . .	342
8-21	<b>db2trc off</b> command . . . . .	342
8-22	<b>Format db2trc trace file</b> command . . . . .	342
8-23	Sample formatted db2trc report . . . . .	342
8-24	SET CLIENT USERID example . . . . .	343
8-25	SET CLIENT WRKSTNNAME example . . . . .	344
8-26	SET CLIENT APPLNAME example . . . . .	344
8-27	SET CLIENT ACCTNG example . . . . .	344
8-28	Using accounting information for DDF work classification . . . . .	344
8-29	Enclave details showing the Accounting Information field . . . . .	344
8-30	RMF enclave report . . . . .	345
8-31	db2 get cli cfg and accounting CLI settings . . . . .	347
8-32	DRDA network trace and CLI accounting settings . . . . .	347
8-33	DIS THD and accounting information . . . . .	348
8-34	Timeout report when not using CLI user information . . . . .	349
8-35	Timeout report when using CLI custom information . . . . .	350
8-36	Sample procedure for capturing CTRACE information . . . . .	350
8-37	Output of display trace command . . . . .	351
8-38	IPCS primary options menu . . . . .	352
8-39	PCS CTRACE option menu . . . . .	352
8-40	PCS CTRACE reporting options . . . . .	353
8-41	Trace initialization message . . . . .	353
8-42	Sample formatted trace output . . . . .	353
8-43	Formatted report from IPCS for packet trace . . . . .	354
8-44	Summary of the IP trace CSV format file using the inhouse tool . . . . .	355
8-45	Analysis of DRDA packets in a flat file . . . . .	360
8-46	DRDA SECCHK Non encrypted example . . . . .	362
8-47	DRDA SECCHK Encrypted enabled example . . . . .	363
8-48	START TRACE command syntax, partial . . . . .	364
8-49	Starting IFCID 180 . . . . .	365
8-50	START TRACE output example . . . . .	365
8-51	<b>DIS TRACE</b> command example . . . . .	365
8-52	<b>STOP TRACE</b> command example . . . . .	365
8-53	Switch SMF command . . . . .	366
8-54	Identify SMF dump data set from SMF Dump job . . . . .	366
8-55	Extract DB2 SMF records from SMF dump dataset . . . . .	366
8-56	SYS1.PROCLIB(GTFDRDA), a GTF proclib example . . . . .	367
8-57	SYS1.PARMLIB(GTFDRDA), GTF option example . . . . .	367
8-58	Start GTF system log output . . . . .	367
8-59	Start GTF: Answer to message AHL125A . . . . .	368
8-60	START TRACE with destination GTF . . . . .	368
8-61	STOP GTF trace . . . . .	368
8-62	Use of LOCATION for starting tracing for non z/OS clients . . . . .	370
8-63	OMEGAMON PE Accounting Trace report—short . . . . .	372



8-64	OMEGAMON PE Accounting Trace report —long	373
8-65	OMEGAMON PE report example: Rows involved in multi rows operations	378
8-66	Sample PL/I program: multi row insert	378
8-67	OMEGAMON PE trace report example, IFCID(192).	380
8-68	OMEGAMON PE IFCID 192 report example	381
8-69	GTF Trace header extract of hlq.SDSNMACS(DSNDQWGT)	381
8-70	GTF trace extract	383
8-71	GTF trace header extract of hlq.SDSNMACS(DSNDQW02), IFCID 180	384
8-72	JCL sample: calling IFCID 180 formatting tool	386
8-73	IFCID 180 formatting example, DATA=N option	386
8-74	IFCID 180 formatting example, DATA=Y option	387
B-1	Running the configuration and installation script in AIX	407
C-1	SQL procedure CALL	420
C-2	Creating and registering an XA datasource	424
C-3	Sample JDBC application running an XA transaction using the JDBC XA API	426
C-4	XAUtil class	428
C-5	Properties file XADatasource1_T4S390	439
C-6	Using Progressive Streaming: XMLTest_RedJava	439
D-1	Korn shell script: executing a DB2 query from UNIX	444
D-2	Korn shell: executing a query script in the background	446
D-3	JCL for execution of REXX parser of GTF containing IFCID 180.	446
D-4	REXX code: parsing of GTF trace and IFCID 180	447

Archived

# Tables

1-1 Terminology table . . . . .	4
1-2 Conversation of AR and AS . . . . .	12
1-3 Components of the DRDA process model . . . . .	13
2-1 Recent history of DB2 client products . . . . .	41
2-2 IBM Data Server Drivers and Clients comparison . . . . .	45
2-3 Replacing DB2 Connect Server with IBM Data Server Drivers or Clients . . . . .	48
2-4 IBM Data Server Client Packages: Latest downloads (V9.7) . . . . .	51
3-1 SYSIBM.LOCATIONS . . . . .	87
3-2 SYSIBM.IPNAMES . . . . .	88
3-3 SYSIBM.USERNAMES . . . . .	89
3-4 DSNZPARM parameters . . . . .	90
3-5 DDF work classification attributes . . . . .	112
3-6 Commands to connect 2-tier to DB2 for z/OS and the information required . . . . .	117
3-7 Commands to connect 3-tier to DB2 for z/OS and the information required . . . . .	119
3-8 Commands to connect 2-tier to DB2 data sharing group and the information required . . . . .	120
3-9 Commands to connect 3-tier to DB2 data sharing group and the information required . . . . .	122
3-10 Connecting DB2 for z/OS to DB2 for LUW . . . . .	123
4-1 Security options for DB2 for z/OS . . . . .	130
5-1 Type 1 and Type 2 CONNECT statements . . . . .	189
5-2 db2cli.ini and db2dsdriver.cfg configuration parameters . . . . .	207
5-3 Comparing pureQuery dynamic and static SQL . . . . .	215
5-4 Comparing limited block FETCH and multi-row FETCH . . . . .	219
5-5 Type 4 driver properties . . . . .	222
5-6 Native versus external procedures . . . . .	224
5-7 Comparison across requesters . . . . .	230
5-8 Fetch/insert feature support by client/driver . . . . .	231
6-1 Server list and calculated ratio . . . . .	242
6-2 The numbers of active connections . . . . .	242
6-3 Recommendation for common deployment . . . . .	251
6-4 Recommended settings for the non-Java-based IBM Data Server Driver . . . . .	254
6-5 Recommended settings for DB2 Connect server configuration . . . . .	258
6-6 Test results after D9C1 failed without Sysplex Distributor . . . . .	261
6-7 Test results after D9C1 failed with Sysplex Distributor . . . . .	264
7-1 Summary of DSNZPARM parameters affecting DBATs . . . . .	271
7-2 Fields affected by roll up for distributed and parallel tasks . . . . .	275
7-3 ACCUMUID acceptable values . . . . .	276
7-4 ACCUMACC and effects on some accounting fields . . . . .	278
7-5 SYSPLAN columns of interest for distributed workloads . . . . .	309
7-6 SYSPACKAGES columns of interest for distributed workload . . . . .	310
8-1 Traces available on distributed components . . . . .	325
8-2 Some DRDA command . . . . .	336
8-3 Trace level options . . . . .	338
8-4 Allowable constraints for each trace type . . . . .	369
8-5 Most common allowable destinations for each trace type . . . . .	372
8-6 Extract of constants definitions, z/Architecture® Reference Summary . . . . .	382
8-7 Extract of work qualifiers and their abbreviations . . . . .	388
A-1 DB2 9 current DRDA-related APARs . . . . .	396

Archived

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX®	IBM®	System i®
CICS®	Informix®	System z10™
Cognos®	InfoSphere™	System z9®
DataPower®	iSeries®	System z®
DB2 Connect™	Language Environment®	Tivoli®
DB2 Universal Database™	OMEGAMON®	VTAM®
DB2®	OS/390®	WebSphere®
developerWorks®	Parallel Sysplex®	z/Architecture®
Distributed Relational Database Architecture™	RACF®	z/OS®
DRDA®	Redbooks®	z/VM®
HiperSockets™	Redbooks (logo)  ®	z9®
i5/OS®	RETAIN®	zSeries®
	SecureWay™	

The following terms are trademarks of other companies:

Cognos, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Hibernate, Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

J2EE, Java, JDBC, JDK, JRE, JVM, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ESP, Excel, Microsoft, MS, SQL Server, Visual Basic, Visual Studio, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

Distributed Relational Database Architecture™ (DRDA®) is a set of protocols that permits multiple local and remote database systems and application programs to work together. Any combination of relational database management products that use DRDA can be connected to form a distributed relational database management system. DRDA coordinates communication between systems by defining what can be exchanged and how it must be exchanged.

DB2® for z/OS® Distributed Data Facility (DDF) is a built-in component that provides the connectivity to and from other servers or clients over the network. DDF is a full-function DRDA-compliant transaction monitor which, equipped with thread pooling and connection management, can support very large networks. Different z/OS workload management priorities can be assigned to different, user-specified classes of DDF-routed application work.

In this IBM® Redbooks® publication, we describe how to set up your DDF environment and how to deploy the DDF capabilities in different configurations, including how to develop applications that access distributed databases.

We also describe a set of more advanced features, such as thread pooling and high availability distributed configurations, in a DB2 data sharing environment, as well as the traces available to do performance monitoring and problem determination.

In summary, we show how a high-volume, highly available transactional application can be successfully implemented with a DB2 for z/OS data server accessed by all types of application servers or clients running on the same or different platform.

## The team that wrote this book

This book was produced by a team of specialists from around the world working in Silicon Valley Laboratory, San Jose, California.

**Paolo Bruni** is an Information Management software Project Leader at the International Technical Support Organization based in Silicon Valley Lab, San Jose. He has authored several IBM Redbooks about DB2 for z/OS and related tools, and has conducted DB2 workshops and seminars worldwide. During Paolo's many years with IBM, both in development and in the field, his work has been mostly related to database systems.

**Nisanti Mohanraj** is a Software Engineer in the DB2 for z/OS Distributed Development Team in the IBM Silicon Valley Lab. She has 7 years of experience in distributed database connectivity and DRDA and 9 years overall in DB2 development. Nisanti holds a Master's Degree in Computer Science from the University of Virginia.

**Cristian Molaro** is an independent consultant and DB2 instructor based in Belgium. He is IBM Data Champion and an IBM Certified DBA and Application Developer for DB2 for z/OS V7, V8, and V9. His main activity is linked to DB2 for z/OS administration and performance. Cristian is co-author of the IBM Redbooks *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637 and *50 TB Data Warehouse Benchmark on IBM System z*, SG24-7674. He holds a Chemical Engineer degree and a Master in Management Sciences. He can be reached at [cristian@molaro.be](mailto:cristian@molaro.be).

**Yasuhiro Ohmori** is an IT Specialist with IBM Japan Systems Engineering Co., Ltd. (ISE) under GTS in Japan. He has more than 7 years of experience in technical support for DB2 for z/OS. Yasuhiro has worked with several major customers in Japan implementing DB2 for z/OS and has conducted workshops for IBMers in Japan. His areas of expertise include DB2 for z/OS, DRDA implementation, DB2 Connect™, and related topics.

**Mark Rader** is a Consulting IT Specialist with IBM Advanced Technical Support (ATS), located in Chicago. He has 25 years of technical experience with IBM, including large systems, communications, and databases. He has worked primarily with DB2 for MVS, OS/390®, and z/OS for more than 20 years, and has specialized in data sharing, performance, and related topics for the past 9 years. Mark has been in ATS for the last 8 years.

**Rajesh Ramachandran** is a Senior Software Engineer in IBM System z® e-Business Services. He currently works in the Design Center in Poughkeepsie as an IT Architect and DB2 assignee. He has 12 years of experience in application development on various platforms, which include z/OS, UNIX®, and Linux® utilizing COBOL, Java™, CICS®, and Forte.



*The authors in SVL. From left to right: Yasuhiro, Nisanti, Cristian, Paolo, Mark, and Rajesh*

Thanks to the following people for their contributions to this project:

Rich Conway  
Roy Costa  
Bob Haimowitz  
Emma Jacobs  
International Technical Support Organization

Atkins Chun  
Jaijeet Chakravorty  
Margaret Dong  
Shivram Ganduri  
Sherry Guo  
Jeff Josten  
Keith Howell  
Gopal Krishnan  
Roger Miller



Todd Munk  
Jim Pickel  
Manish Sehgal  
Hugh Smith  
Bart Steegmans  
Derek Tempongko  
Anil Varkhedi  
Daya Vivek  
Dan Weis  
Sofilina Wilhite  
Paul Wilms  
IBM Silicon Valley Lab

Kanchana Padmanabhan  
IBM Glendale

Toshiaki Sota  
IBM Japan Systems Engineering

Gus Kassimis  
IBM Software Group, Raleigh

Brent Gross  
Anju Kaushik  
Melanie Stopfer  
IBM Toronto Lab

Rick Butler  
Bank of Montreal (BMO) Financial Group, Toronto

Thanks to the authors of the first edition of this book, *Distributed Functions of DB2 for z/OS and OS/390*, SG24-6952-00, published in June 2003:

Bart Steegmans  
Neale Armstrong  
Cemil Cemiloglu  
Srirengan Venkatesh Kumar  
Satoru Todokoro

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

# Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes  
for SG24-6952-01  
for DB2 9 for z/OS: Distributed Functions  
as created or updated on May 26, 2011.

## July 2009, First Edition

This revision of this First edition, published July 2009, reflects the addition, deletion, or modification of new and changed information described below. Change bars mark meaningful changes. Minor typographical corrections might not have change bars.

## September 2009, First Update

This revision reflects the addition, deletion, or modification of new and changed information described below.

### New information

- ▶ Added clarification in 5.8.2, “Using the IBM non-Java-based Data Server Drivers/Clients to enable direct XA transactions” on page 227.
- ▶ Added APARS in Appendix A, “DRDA-related maintenance” on page 395.
- ▶ Added Example C-4 on page 428 and Example C-5 on page 439 in Appendix C, “Sample applications” on page 419.

### Changed information

- ▶ Updated text in “Preface” on page xxv to reflect the correct title of the previous edition.
- ▶ Updated text in 1.7.3, “Transaction pooling” on page 25.
- ▶ Updated text in 3.1.3, “Basic TCP/IP setup” on page 71.
- ▶ Updated text in Table 5-7 on page 230.
- ▶ Updated text in three bullets in 5.9, “Remote application recommendations” page 230.
- ▶ Updated Table 6-3 on page 251.
- ▶ Updated text in 8.2.1, “TCP/IP packet tracing on z/OS” on page 352.
- ▶ Updated APARS in Appendix A, “DRDA-related maintenance” on page 395.

## May 2011, Second Update

This revision reflects the addition, deletion, or modification of new and changed information described below.

### **Changed information**

- Corrected text of reference in 5.7.6, “Multi-row MERGE” on page 221 and text in Example 5-31 on page 221.



# Part 1

## **Distributed database architecture and configurations**

In this part we introduce the concepts and protocols of DRDA and describe the layout and the components of the possible configurations where DB2 for z/OS can play a client or server role.

This part contains the following chapters:

- ▶ Chapter 1, “Architecture of DB2 distributed systems” on page 3
- ▶ Chapter 2, “Distributed database configurations” on page 33

Archived

# Architecture of DB2 distributed systems

In this chapter we discuss distributed data topology and provide a brief introduction to the Distributed Relational Database Architecture (DRDA) and its general implementation in the family of DB2 products. We provide some details about the Distributed Data Facility (DDF), the DB2 for z/OS component that deals with data access from remote applications as well as the various clients that can connect to DB2 for z/OS.

Lastly, we mention (but do not use in this book) the Federated Data Server products that allow you to perform distributed requests.

This chapter contains the following sections:

- ▶ “Before you start” on page 4
- ▶ “Distributed data access topology” on page 4
- ▶ “DRDA” on page 9
- ▶ “Implementation of DRDA in the DB2 family” on page 14
- ▶ “Implementation of DRDA by non-IBM products” on page 16
- ▶ “DB2 for z/OS Distributed Data Facility architecture” on page 17
- ▶ “Federated data support” on page 26

## 1.1 Before you start

In this IBM Redbooks publication we describe many features and functions of products that are constantly undergoing changes. At the time of writing, we used DB2 9 for z/OS and DB2 for Linux, UNIX, and Windows®, 9.5 FixPack 3. For certain configurations, we used the newest release of DB2 for LUW 9.7 and we will explicitly mention that.

The hardware and software configuration used during this project is described in Appendix B.2.2, “TRADE installation” on page 405.

### Terminology

DB2 for z/OS used in this book generally refers to DB2 9 for z/OS unless DB2 for z/OS V8 is explicitly mentioned. Table 1-1 lists the abbreviations used throughout this publication.

Table 1-1 Terminology table

Official term	Term used in this book
DB2 for Linux, UNIX and Windows 9.5 FixPack 3	DB2 for LUW 9.5 FP3
DB2 Version 9.1 for z/OS	DB2 9 for z/OS
DB2 for z/OS Version 8	DB2 for z/OS V8
IBM Data Server Driver for JDBC™ and SQLJ, Type 4 Connectivity	Type 4 driver
IBM Data Server Driver for JDBC and SQLJ, Type 2 Connectivity	Type 2 driver
IBM Data Server Driver for ODBC and CLI	CLI Driver

## 1.2 Distributed data access topology

Before discussing distributed data topology, we need to introduce two important concepts, unit of work and two-phase commit.

### Unit of work

A unit of work (UOW) is a single logical transaction. It consists of a sequence of SQL statements that are either all successful, or considered as a whole to be unsuccessful. It maintains data integrity within a transaction.

Figure 1-1 on page 5 illustrates two sequences of SQL statements that are executed. In an environment where the UOW concept is not supported (on the left in the figure), if one SQL statement fails, the data updated by previously executed SQL statements in the transaction remains. SQL3 fails but data updated by SQL1 and SQL2 remains updated. On the other hand, in a UOW-supported environment (on the right in the figure), if one statement fails (SQL3), all the updates are discarded and affected data is reinstated to the state before the update (affected by SQL1 and SQL2). This way, data integrity at the transaction level can be maintained.



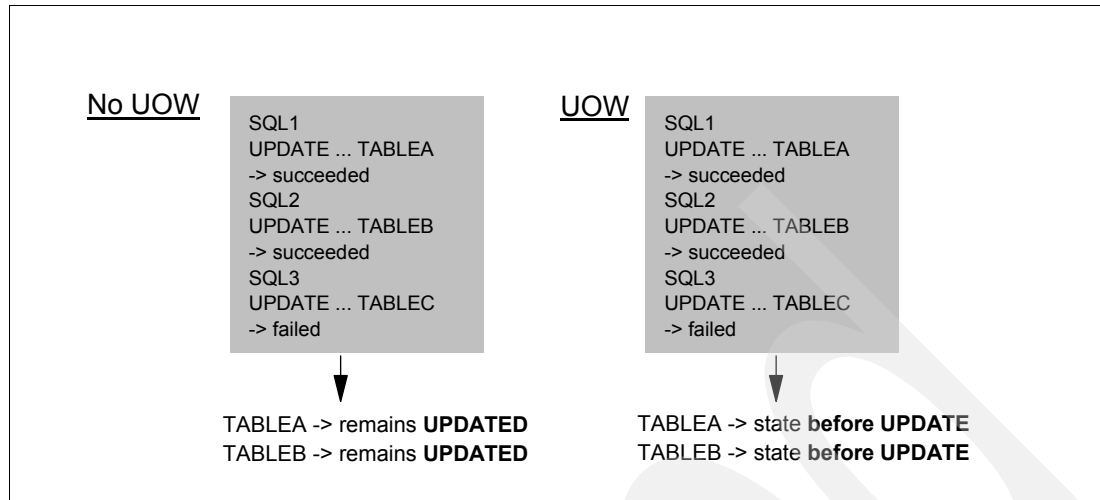


Figure 1-1 Unit of work concept

## Two-phase commit protocol

The second key concept to understand is the two-phase commit protocol. Two-phase commit enables the UOW concept across multiple DBMSs.

Continuing to use the scenario from the previous section, two-phase commit involves a coordinator and one or more participants. The coordinator coordinates the UOW. The coordinator itself may also be a participant. Figure 1-2 shows the components and message flows between the partners when the presumed abort protocol (discussed below) is used.

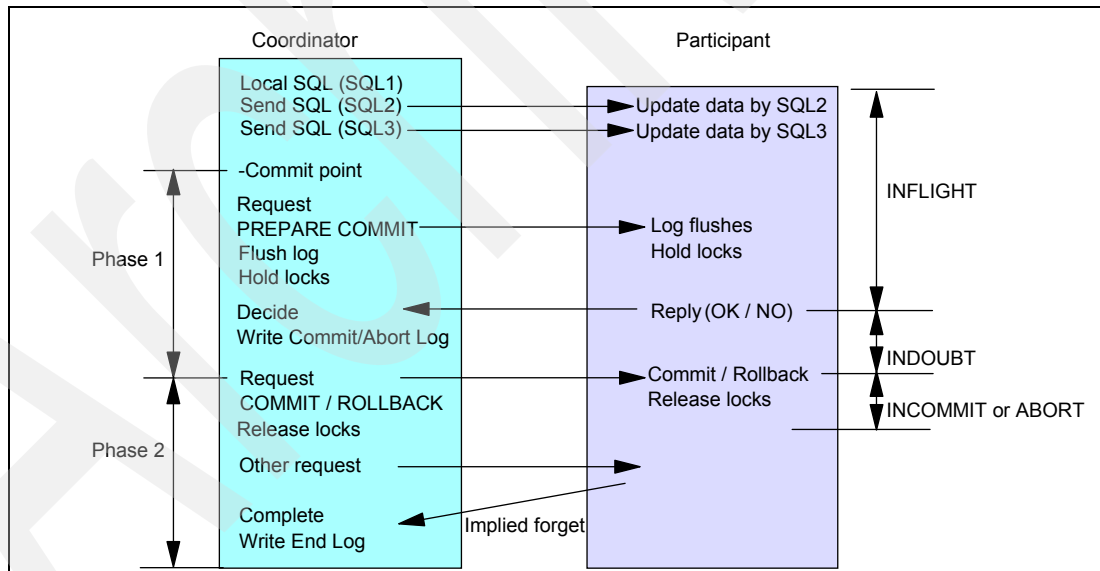


Figure 1-2 Two-phase commit protocol

At first, all SQL requests are sent to the database (remote or local), and the data is updated in the database. Now the application decides it is time to commit. The coordinator sends a “PREPARE to COMMIT” request to (all) participants. After receiving the request, a participant prepares for the COMMIT. This includes flushing log records (and logging the commit phase 1 record) and keeping locks. When this is done, the participant replies to the coordinator that it is ready to commit. (Preparation is also done by the coordinator.) When the coordinator has received the replies from all participants, the coordinator can go ahead with the commit and

write the commit log record. The coordinator sends the COMMIT request to all participants. After receiving the request, the participant commits the update (logging a begin phase 2 record), releases the locks, and optionally sends a reply to the coordinator. When the coordinator receives the replies from all the participants (either through an explicit confirmation message, or the confirmation is implied with the next message), it considers the UOW as completed. This way, the coordinator manages UOWs across participants.

The point from when the application starts the commit processing to the time the actual COMMIT request is issued, is called *phase 1*. The time the COMMIT request is issued until the end of the transaction is called *phase 2*. That is why this protocol is called a two-phase commit protocol.

A transaction can be in any of the following states: INFLIGHT, INDOUBT, INCOMMIT or ABORT, according to the phase to which it belongs (see Figure 1-2 on page 5). When the participant encounters an unexpected problem when it is in doubt (for example, after the system goes down due to power loss, and so forth), it can ask the coordinator the state of the transaction, and recover according to the information provided by the coordinator. Thus, a sequence of SQL statements across multiple DBMSs can be managed as a single consistent transaction, which enables distributed units of work in distributed data environments.

There are three well-known variants of two-phase commit protocols, *presumed nothing*, *presumed abort*, and *presumed commit*. Presumed nothing is the basic two-phase commit protocol that is only supported by SNA and not supported by TCP/IP, which is the recommended network protocol. Presumed Commit is a variation that is not implemented by DRDA, so we limit our discussion to the Presumed Abort protocol.

Presumed Abort is an enhancement to the basic two-phase commit protocol that is designed to reduce the number of messages exchanged and the number of times that log records are written. When using the Presumed Abort variation, a coordinator can forget about the transaction when it decides to abort. Whenever the participant asks about a transaction for which the coordinator does not have information, it replies that the transaction is aborted. In addition, participants do not have to reply to rollback requests from the coordinator. This variation reduces writing logs by the coordinator, and also reduces the number of messages exchanged when the transaction is aborted.

**Important:** While DB2 for z/OS requesters always use two-phase commit when communicating with a DB2 for z/OS server, most other requesters default to one-phase commit, and require the use of global XA transactions to ensure that two-phase commit is used. We will discuss this in 5.8, “XA transactions” on page 225.

Let us now have a closer look at the different types of distributed data access topology.

### 1.2.1 Remote request

This is the basic remote access. An application (SQL) request accesses a single remote data management system (DBMS), and each transaction (UOW) consists of only one SQL statement. An SQL COMMIT statement is invoked either explicitly or implicitly. The remote request concept, and a sample configuration, is shown in Figure 1-3 on page 7. Through this section, a UOW is represented by a shaded rectangle. For example, one SQL statement and a COMMIT statement are included in each UOW in the figures.

**Note:** The term DBMS in this and the following sections refers to an entire DB2 for z/OS subsystem or data sharing group (in a distributed environment often addressed through its location name), or DB2 LUW database server.

A remote request has the following characteristics:

- ▶ One request per unit of work
- ▶ One DBMS per unit of work
- ▶ One DBMS per request

An application can access multiple databases. However, only one database is accessed at a time and only one SQL statement is included in a single UOW. In this case, the result of one SQL statement will not affect the result of another SQL statement.

Figure 1-3 shows a banking deposit application. The customer can deposit money to either the checking or the savings account, using an ATM terminal. Whether the deposit into his checking account is successful does not affect the deposit into the savings account.

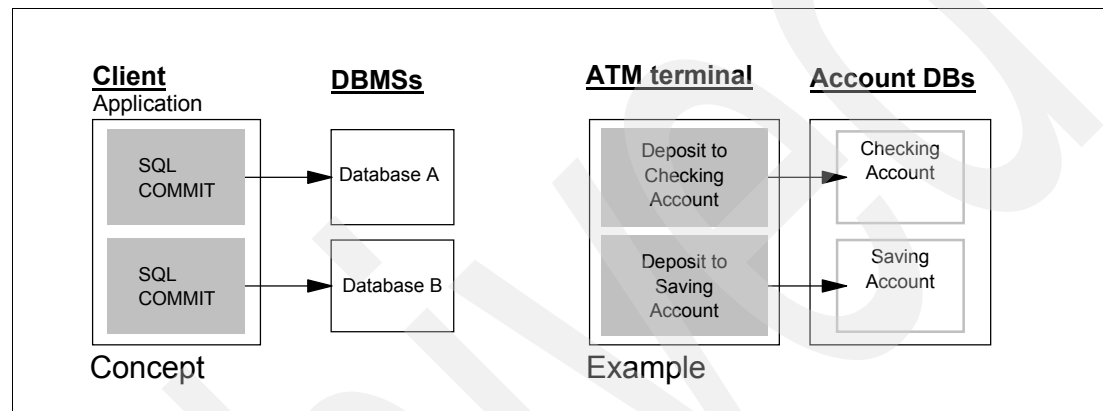


Figure 1-3 Remote request

## 1.2.2 Remote unit of work

A *Remote unit of work (RUW)* is the next level of distributed data access. A remote unit of work lets an application program read or update data on one DBMS per unit of work.

Remote unit of work has following characteristics:

- ▶ Multiple requests per unit of work
- ▶ One DBMS per unit of work
- ▶ One DBMS per request
- ▶ Application can initiate commit processing
- ▶ Commit scope is a single DBMS

Figure 1-4 on page 8 shows the concept and an example. Although two databases are accessible, only one DBMS can be included in one UOW. A UOW can include multiple SQL statements.

Figure 1-4 on page 8 shows a deposit transfer application where a UOW is represented by a shaded rectangle. Assume that a customer has two sets of checking and savings accounts in two branches. A customer can transfer money from his checking account to his savings account as long as it is within the same branch. The data of both the checking and the savings account is maintained in a single database within each branch. In this case, withdrawing from the checking account and depositing into the savings account (of the same branch office) must be treated as a single event, in other words, a transaction, a single unit of work. However, the two transactions for two branches do not affect each other.

This type of distributed data access is defined in DRDA level-1.

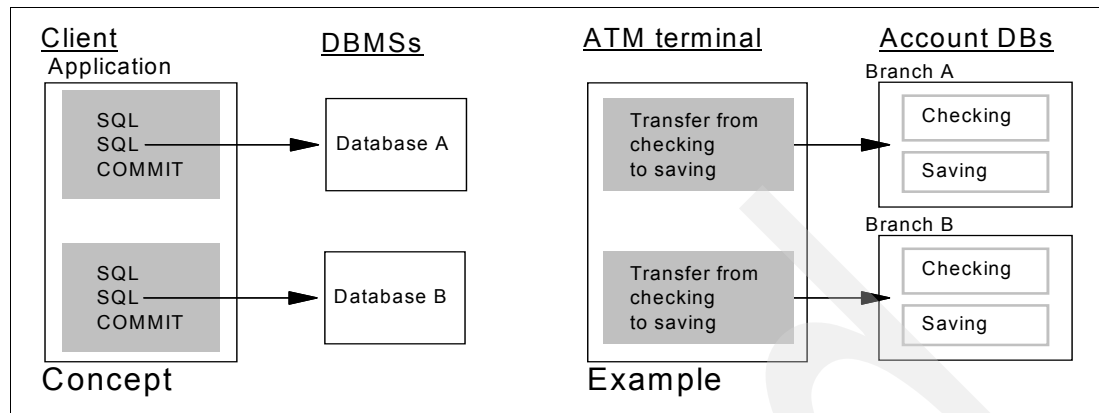


Figure 1-4 Remote unit of work

### 1.2.3 Distributed unit of work

A *distributed unit of work (DUW)* involves more than one DBMS within a single unit of work. Within one unit of work, an application can direct SQL requests to multiple DBMSs. In case the application is aware of the database distribution, it indicates the target DBMS when it executes the SQL statement. In DB2 this is done using the `CONNECT TO` statement. However, you can also achieve some level of location transparency by creating an alias. This means that the application is unaware of where the data actually resides. Note that location transparency is not a DUW concept. It is a general distributed database concept that applies to all levels of data distribution. When using a DUW, all objects referenced in a single SQL statement are constrained to be in a single DBMS.

Distributed unit of work has the following characteristics:

- ▶ Several DBMSs per unit of work.
- ▶ Multiple requests per unit of work.
- ▶ One DBMS per request.
- ▶ Application can initiate commit processing.
- ▶ Commit coordination across multiple DBMSs.

Two-phase commit support is essential to this implementation.

Figure 1-5 on page 9 shows the concept and an example. SQL statements are within one UOW. A UOW is represented by a shaded rectangle. If one of the statements fails, all affected data (in all DBMSs involved in the UOW) is reset to the state it was before the UOW started.

Figure 1-5 on page 9 shows the case where the customer asks for a transfer of funds from the account in bank A to the account in bank B. In this case, the withdrawal in bank A and the deposit in bank B must not be treated as different events. They should be treated as parts of a single transaction.

By supporting the distributed unit of work concept, we can implement a transaction concept across multiple DBMSs. This type of transaction is defined in DRDA level-2.

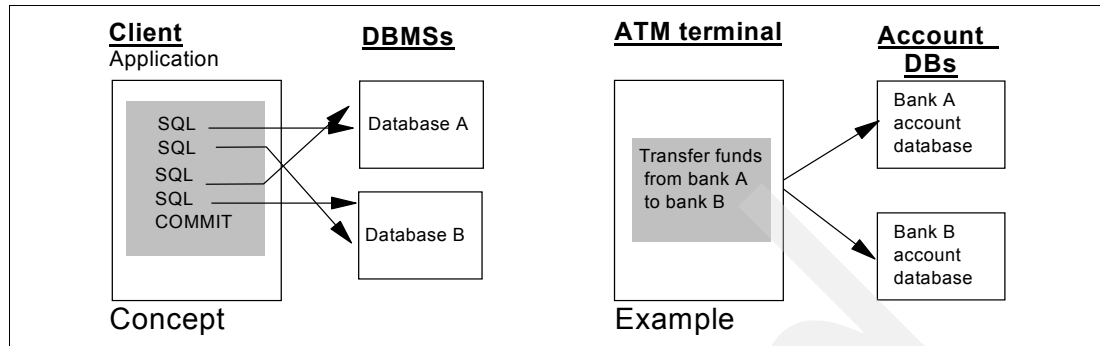


Figure 1-5 Distributed unit of work

## 1.2.4 Distributed request

The fourth and last distributed data access is the *distributed request*. It allows users and applications to submit SQL statements that reference multiple DBMSs in a single SQL statement. For example, you can execute a join between a table in system A and another table in system B. The concepts and an example are shown in Figure 1-6. Note that you need the use the IBM InfoSphere™ Federation Server products (mentioned in 1.8.1, “IBM InfoSphere Federation Server” on page 28 and 1.8.2, “IBM InfoSphere Classic Federation Server for z/OS” on page 29) to support distributed requests.

A distributed request has the following characteristics:

- ▶ Several DBMSs per unit of work
- ▶ Multiple requests per unit of work
- ▶ Multiple DBMSs per request
- ▶ Application can initiate commit processing
- ▶ Commit coordination across multiple DBMSs

Figure 1-6 show that a customer can request the maximum checking balance across all accounts in all banks where he has an account.

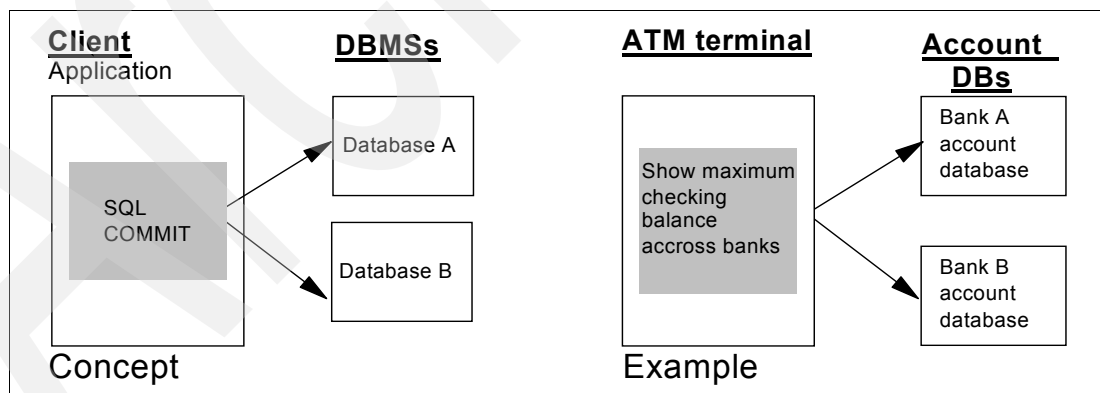


Figure 1-6 Distributed request

## 1.3 DRDA

A common protocol that is independent of the underlying RDBMS architecture and operating environments is necessary to access a diverse set of RDBMSs. The IBM DB2 distributed database functionality is based on DRDA. DRDA is an open, vendor-independent architecture

for providing connectivity between a client and database servers. It was initially developed by IBM and then adopted by The Open Group<sup>1</sup> as an industry standard interoperability protocol. Connectivity is independent from not only hardware and software architecture but also vendors and platforms (see Figure 1-7). Using DRDA, an application program can access various databases that support DRDA, using SQL as a common access method.

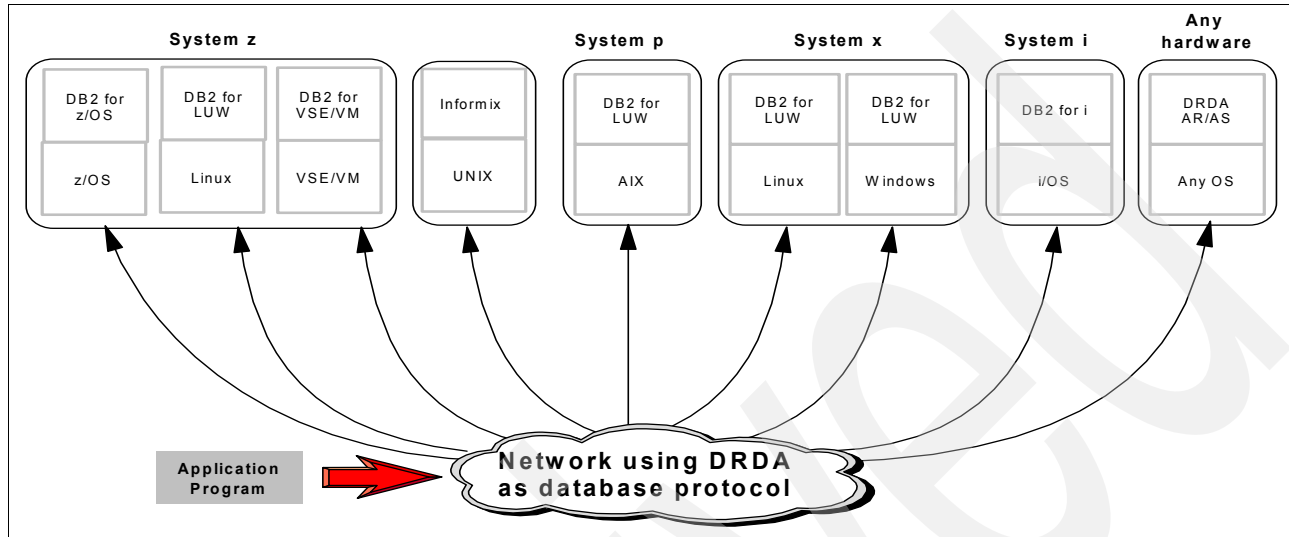


Figure 1-7 DRDA network

The major characteristics of DRDA are as follows:

- ▶ Support for any SQL dialect, static (previously bound) and dynamic.
- ▶ Automatic data transformation through 'receiver makes right' philosophy.
- ▶ UOW support including 2-phase commit, recovery, and multi-site updates.
- ▶ Stored procedure support
- ▶ Superior performance, scalability, and availability
- ▶ Support for enhanced security mechanisms including data encryption, trusted context, and Kerberos
- ▶ Support for an XA distributed transaction processing (DTP) interface

DRDA is defined by using different levels so that vendors can implement their applications step by step, depending on the level of DRDA. Some of the functions introduced in DB2 9 and the latest level of DRDA are as follows:

- ▶ XML extensions  
DRDA supports new XML types. XML data are treated similar to LOB data from a DRDA perspective and flowed through EXTDTAs
- ▶ Dynamic data format  
DRDA allows the in-lining of small LOBs in QRYDTA and the chunking of large LOBs into progressive references that can be retrieved through the GETNXTCHK command.
- ▶ Improved package management  
DRDA is enhanced to support remote bind COPY and bind DEPLOY so a package can be copied or deployed (no change in original bind options) from a source server to target.
- ▶ Enhanced security  
Support for trusted context and 256-bit Advanced Encryption Standard (AES) encryption.

<sup>1</sup> The Open Group Web page is <http://www.opengroup.org/dbiop/>

### 1.3.1 Functions and protocols of DRDA

For the purposes of this book, we are only giving a brief introduction to DRDA. DRDA is fully documented in documents available online from the Open Group:

- ▶ *DRDA V4, Vol. 1: Distributed Relational Database Architecture*  
<http://www.opengroup.org/onlinepubs/9699939399/toc.pdf>
- ▶ *DRDA V4, Vol. 2: Formatted Data Object Content Architecture*  
<http://www.opengroup.org/onlinepubs/9699939299/toc.pdf>
- ▶ *DRDA V4, Vol. 3: Distributed Data Management Architecture*  
<http://www.opengroup.org/onlinepubs/9699939199/toc.pdf>

DRDA is a set of protocols and functions providing connectivity between applications and database management systems. See Figure 1-8.

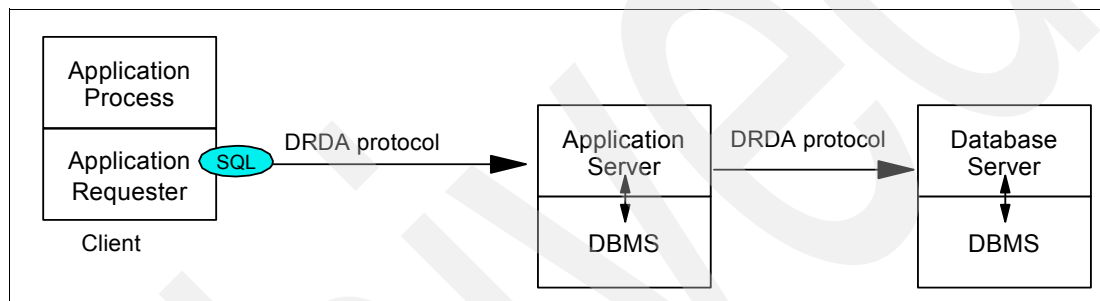


Figure 1-8 Functions and protocols used by DRDA

The following function types are provided:

- ▶ **Application requester**  
Application requester (AR) functions support SQL and program preparation services from applications. The AR is SQL neutral, so it can accept SQL that is supported on any DBMS.
- ▶ **Application server**  
Application server (AS) functions support requests that application requesters have sent, and route requests to database servers by connecting as an application requester.
- ▶ **Database server**  
Database server (DS) functions support requests from application servers. They support the propagation of special register settings and can forward SQL requests to other database servers.

The following protocols are provided:

- ▶ **Application Support Protocol**  
Application Support Protocol provides connection between application requesters (AR) and application servers (AS).
- ▶ **Database Support Protocol**  
Database Support Protocol provides connections between application servers (AS) and database servers (DS). Prior to executing any SQL statements at a database server, special register settings set by the application are propagated to the database server.

### 1.3.2 Conversations between the AR and AS

Connections are managed by the application. The roles of the two functions during their communication is shown in Table 1-2. Equivalent functionality exists for communications between the application server and the database server.

Table 1-2 Conversation of AR and AS

Application requester (AR)	Application server (AS)
Initiates connection to the AS	Listens for DRDA requests
Generates DRDA request	Parses DRDA request
Sends requests to the AS	Invokes RDBMS
Receives replies from the AS	Generates DRDA replies
Parses replies from the AS	Sends replies to the AR
Terminates connection	Connection termination implies rollback

### 1.3.3 Building blocks of DRDA

DRDA requires the following architectures:

- ▶ Distributed Data Management (DDM)  
The DDM architecture provides the command and reply structure used by the distributed databases.
- ▶ Formatted Data Object Content Architecture (FD:OCA)  
The FD:OCA provides the data definition architectural base for DRDA.
- ▶ Character Data Representation Architecture (CDRA)  
CDRA provides the consistency of character data across the multiple platforms.
- ▶ Communication protocol  
SNA and TCP/IP are available.

### 1.3.4 The DRDA process model

Figure 1-9 on page 13 shows the DRDA process model and the function of each process (called manager in DDM terminology) and objects.



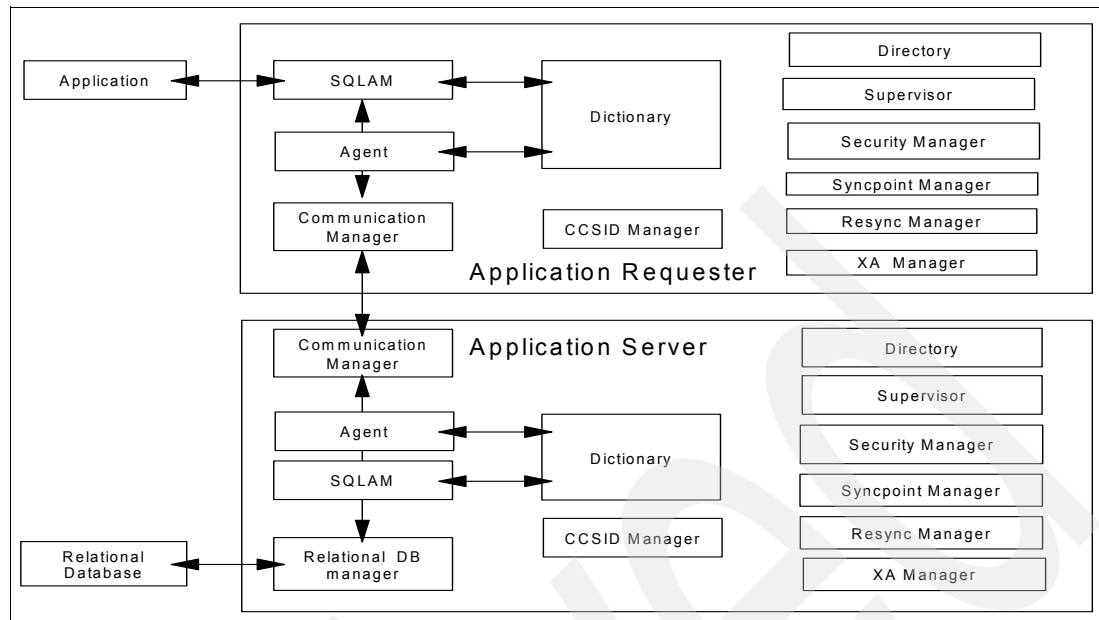


Figure 1-9 DRDA process model

A brief description of the components is given in Table 1-3.

Table 1-3 Components of the DRDA process model

Manager/process/object	Function/description
SQL Application Manager (SQLAM)	Represents the application to the remote relational database manager; handles all DRDA flows.
Directory	Maps the names of instances of manager objects to their locations.
Dictionary	A set of named descriptions of objects.
Communications Manager	Provides conversational support for the agent in an AR or AS through SNA or TCP/IP.
Agent	In an AR, the agent interfaces with the SQLAM to receive requests and pass back responses. In an AS, the agent interfaces with managers in its local server to determine where to send the command to be processed, to allocate resources and to enforce security.
Supervisor	Manages a collection of managers within a particular operating environment.
Security Manager	Participates in user identification and authentication, and ensures the access of the requester.
Resynchronization Manager	A system component that recovers protected resources when a commit operation fails.
Syncpoint Manager	A system component that coordinates commit and rollback operations among various protected resources.
Relational Database Manager	Controls the storage and integrity of data in a relational database. DRDA provides no command protocol or structure for relational database managers.

Manager/process/object	Function/description
CCSID Manager	Allows the specification of a single-byte character set CCSID to be associated with character typed parameters on DDM commands and DDM reply messages.
XA Manager	Provides a data stream architecture that will allow the application requester and server to perform the operations involved in protecting a resource.

## 1.4 Implementation of DRDA in the DB2 family

DRDA is implemented throughout the IBM relational data base systems as well as in relational database management systems provided by companies other than IBM. In this section we briefly describe how DRDA is implemented and used in the IBM products.

### 1.4.1 DB2 for z/OS

Both Application Server and Application Requester functions are implemented as standard functions of DB2 for z/OS.

### 1.4.2 DB2 for i

Both Application Server and Application Requester functions are implemented as standard functions of DB2 for i 6.1 (formerly known as DB2 for i5/OS®).

### 1.4.3 DB2 Server for VSE and VM

The DB2 Server for VSE and VM V7.5 implements a DRDA Application Server. With the DB2 Runtime only Client editions for VM and VSE features that implement the DRDA Application Requester, you can purchase and use only the DB2 client, without having to pay for the database server.

### 1.4.4 IBM Informix Dynamic Server

Starting with Version 11.10, the IBM Informix® Dynamic Server (IDS) can act as a DRDA Application Server. The IBM Data Server Driver for JDBC and SQLJ (described in 1.4.6, "IBM Data Server Driver for ODBC and CLI" on page 15) as well as the IBM Data Server Driver for .NET (described in 1.4.6, "IBM Data Server Driver for ODBC and CLI" on page 15) can be used as DRDA Application Requesters when connecting to the Informix Dynamic Server.

### 1.4.5 DB2 for Linux, UNIX and Windows

On DB2 for Linux, UNIX, and Windows (DB2 for LUW), the DRDA Application Server support is provided as a standard function within the DB2 server engine. On these platforms, the DRDA Application Requester function is an optional feature that can be ordered with the database server, or can also be licensed separately in the form of the DB2 Connect product or any of the IBM Data Server Drivers or Clients listed below.

Here is the list of the IBM Data Server Clients and Drivers:

- ▶ IBM Data Server Client
- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Driver for ODBC and CLI
- ▶ IBM Data Server Driver for JDBC and SQLJ
- ▶ IBM Data Server Driver Package

In addition, a separate product, the DB2 Connect Client, includes all the functionality of IBM Data Server Client plus the capability to connect to midrange and mainframe databases. DB2 Connect capability can be added to any client or driver.

The DB2 Connect Server is considered a DRDA Application Server because it has the ability to perform Connection Concentration and Sysplex Workload Balancing. Although the terms Connect Gateway and Connect Server are used interchangeably, Gateway was never an official term and we will refer to it as DB2 Connect Server. Prior to Version 8.1, DB2 Connect Server was performing protocol conversions (from the DB2 for LUW version of private protocol to DRDA) as well as codepage conversions when connecting to the DB2 for z/OS server. Starting with Version 8.1, these conversions no longer need to be performed as long as the DB2 clients are Version 8.1 or later.

Type 4 driver added sysplex workload balancing for JDBC 2.0 data sources in DB2 for LUW V8 FP10 and for JDBC 1.2 ConnectionManager connections in DB2 for LUW V9. Sysplex workload balancing was added to the non-Java-based client in DB2 LUW 9.5 FP3, eliminating the need for DB2 Connect Servers to act as a middle tier between DB2 clients and DB2 for z/OS servers.

You can run DB2 CLI and ODBC applications against a DB2 database server using the IBM Data Server Client, the IBM Data Server Runtime Client, or the IBM Data Server Driver for ODBC and CLI. However, to compile DB2 CLI or ODBC applications, you need the IBM Data Server Client.

See the DB2 Version 9.5 for Linux, UNIX, and Windows Information Center for an overview of IBM Data Server Clients and Drivers at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.swg.im.dbclient.install.doc/doc/c0022612.html>

### 1.4.6 IBM Data Server Driver for ODBC and CLI

The IBM Data Server Driver for ODBC and CLI (CLI driver) provides runtime support for the c-DB2 CLI application programming interface (API), the ODBC API, the XA API and connecting to databases. We refer to these clients and the .NET support drivers as non-Java based clients.

### 1.4.7 IBM Data Server Driver for JDBC and SQLJ

JDBC is an application programming interface (API) that Java applications use to access relational databases. The IBM Data Server Driver for JDBC and SQLJ provides Type 4 and Type 2 connectivity. To communicate with remote servers using DRDA, the Type 4 driver is used as a DRDA Application Requester. SQLJ provides support for embedded static SQL in Java applications. In general, Java applications use JDBC for dynamic SQL and SQLJ for static SQL.

This driver is also called the JAVA Common Client (JCC) driver and was formerly known as the IBM DB2 Universal Database™ Driver. The DB2 JDBC Type 2 driver for LUW, also called the CLI existing driver, has been deprecated.

We can refer to these clients as Java-based clients.

See the Java application development for IBM data servers section at the DB2 Version 9.5 for Linux, UNIX, and Windows Information Center at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.java.doc/doc/c0024189.html>

### 1.4.8 IBM Data Server Driver Package

In June 2009, IBM announced the IBM Data Server Driver Package, a lightweight deployment solution providing runtime support for applications using ODBC, CLI, .NET, OLE DB, open source, or Java APIs. This replaced the separate drivers formerly available for .NET and OpenSource. All of the IBM Data Server Drivers have a small footprint, are designed to be redistributed by independent software vendors (ISVs), and to be used for application distribution in mass deployment scenarios typical of large enterprises.

### 1.4.9 pureQuery

pureQuery is a high-performance data access platform that makes it easier to develop, optimize, secure, and manage data access. It consists of the following features:

- ▶ Application programming interfaces that are built for ease of use and for simplifying the use of best practices
- ▶ Development tools, which are delivered in Data Studio Developer, for Java and SQL development
- ▶ A runtime, which is delivered in Data Studio pureQuery Runtime, for optimizing and securing database access and simplifying management tasks

IBM Data Studio pureQuery Runtime for z/OS offers a runtime environment that optimizes data access performance of Java applications and stored procedures deployed on the z/OS platform. pureQuery uses the IBM Data Server Driver for JDBC and SQLJ type 4 connectivity to communicate through DRDA to DB2 for z/OS, DB2 LUW and IDS servers. See 5.6, “Developing static applications using pureQuery” on page 211 for details on pureQuery.

## 1.5 Implementation of DRDA by non-IBM products

As mentioned, DRDA is an industry standard initially developed by IBM and adopted by The Open Group as a database interoperability industry standard. There are a number of non-IBM products that have implemented DRDA. Refer to The Open Group Web site for details, at the following Web page:

<http://www.opengroup.org/dbiop/>

## 1.6 DB2 for z/OS Distributed Data Facility architecture

In this section, we describe the Distributed Data Facility (DDF) structure. DDF is the DB2 for z/OS implementation of distributed database access. We describe the network protocols used by DDF:

- ▶ TCP/IP
- ▶ SNA

We talk about inactive connection support provided by DDF and how this facilitates connection pooling and transaction pooling.

### 1.6.1 What DDF is

DB2 for z/OS Distributed Data Facility (DDF) is a built-in component of DB2 for z/OS which provides the connectivity to and from other databases over the network. DDF implements a full DRDA Application Server and Application Requester.

DDF is DB2's transaction manager for distributed database connections. DDF has developed mature thread management strategies to handle thousands of connections that can come from anywhere within the bounds of the network that DB2 is operating in.

DDF runs as an additional address space in the DB2 subsystem. The address space name is *ssid*DIST, where *ssid* is the DB2 subsystem name. DDF is an efficient connection handler. It uses SRBs instead of TCBs, which reduces CPU time. z/OS enclaves are used in exchanging data across address spaces. This enables proper management by Workload Manager (WLM) of the work coming into DB2 through DDF and the possibility of routing DDF work to the zIIP specialty engine.

### 1.6.2 Distributed configurations

Before we discuss the DDF implementation, it is important to have an understanding of how distributed clients can connect to a DB2 for z/OS server.

When DDF was first introduced in DB2 Version 2.2, only another DB2 for z/OS subsystem could communicate through Private Protocol to a DB2 for z/OS server.

With DB2 Version 2.3 (which introduced DRDA RUW support), Version 3 (which introduced DUW support) and the standardization of DRDA as a communication protocol, any DRDA compliant requester could communicate with a DB2 for z/OS server. The popular configuration was clients connecting to DB2 for z/OS through the various editions of DB2 Connect.

Later, connecting to DB2 for z/OS through the WebSphere® Application Server provided developers and IT Architects with an innovative, performance-based foundation to build, reuse, run, integrate, and manage service-oriented architecture (SOA) applications and services.

Today, the most common customer configurations include application servers connecting through one of the IBM Data Server Drivers (either Java-based or non-Java-based) to a DB2 for z/OS server.

We expect the 'gateway' type accesses to become less popular with the Data Servers Drivers offering equivalent and ever improving functions.

Figure 1-10 illustrates the most common ways distributed clients connect to a DB2 for z/OS server.

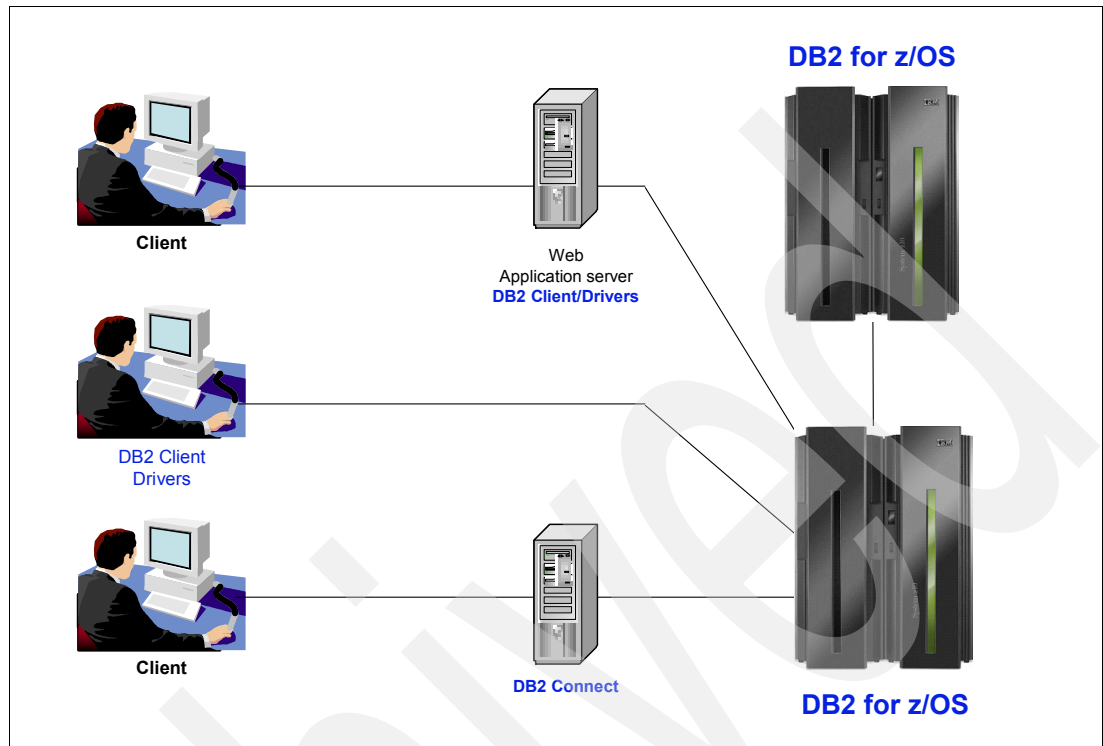


Figure 1-10 Connecting to DDF

### 1.6.3 DDF implementation

An overview of the address spaces used by a DB2 z/OS to DB2 z/OS communication is shown in Figure 1-11 on page 19, where APPL, DBM1, MSTR, DIST, and IRLM stand for application, database service, system service, distributed data function, and lock manager address spaces, respectively. This figure does not show other WLM-managed address spaces that are used for non-native stored procedures and user-defined functions, nor the Admin Scheduler address space new with DB2 9 for z/OS.

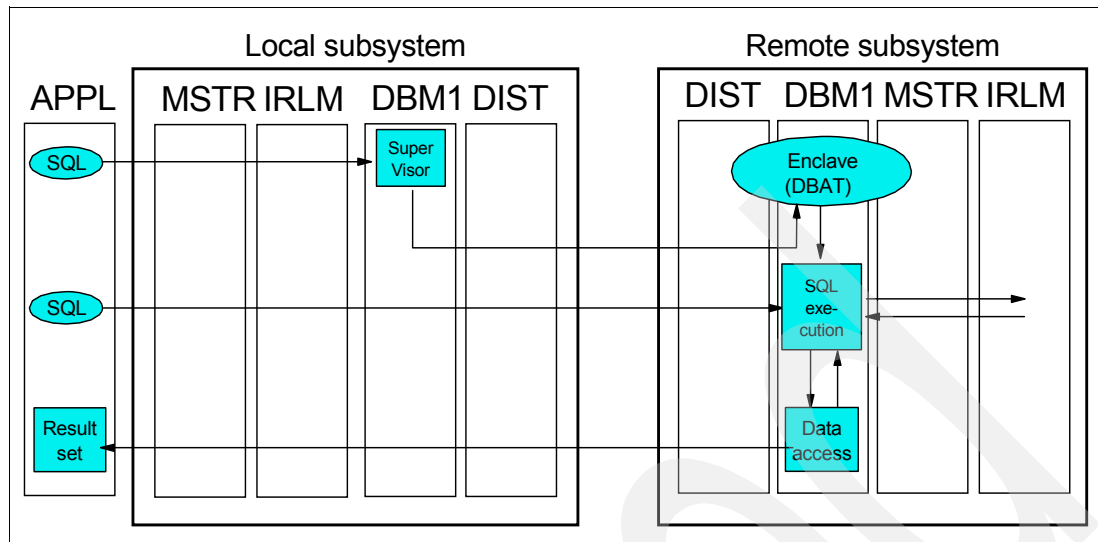


Figure 1-11 Address spaces

Figure 1-11 shows a simplified view of the message exchanges that occur when an application running on DB2 for z/OS requests to access data on remote DB2 for z/OS subsystem.

As mentioned earlier, the DDF functions execute in the ssidDIST address space. Processes running in the DIST address space access the DB2 database services address space (DBM1) using cross memory services and the Shared Memory facility. Cross Memory Services allow synchronous use of data and programs in different address spaces. In DB2 9 for z/OS, most of DDF control blocks are moved above the 2 GB bar. This removes storage constraints for communications buffers and DRDA intensive usage by vendor applications.

With 64 bit, DB2 DDF uses the z/OS Shared Memory Facility to reduce data moves between DBM1 and DDF. Shared memory is a type of virtual storage introduced in z/OS 1.5 that resides above the 2 GB bar and allows multiple authorized address spaces to easily address storage. 64-bit DDF is a performance enhancement for distributed server processing, but it also provides virtual storage constraint relief. No cross memory moves are necessary for the shared blocks, and this storage no longer needs to be allocated in the DBM1 address space below the 2 GB bar. DDF also exploits shared private storage with TCP/IP and Unix System Services for network operations.

In a local DB2 subsystem acting as application requester, as part of the program preparation process, an application program is link edited with a language interface module (for example, DSNELI for TSO). This enables the program to send SQL statements to the local subsystem, making calls to the language interface module. When the statement references a remote location, the request is sent to the remote subsystem through the DDF address space of the local subsystem.

When the first SQL statement from a remote system is received by the DDF address space of the DB2 subsystem acting as application server, a thread is created in the DBM1 address space. This type of DB2 thread is called a Database Access Thread (DBAT)<sup>2</sup>. Otherwise, and most likely, if a thread for a remote connection already exists, but is idle (pooled DBAT), it is reused for the request. The DBAT is implemented through an z/OS enclave, performing work through preemptive SRBs running in the DDF address space. Subsequent SQL statements

<sup>2</sup> DB2 for z/OS associates a thread to the application request for work in the DBM1 address space (SQL accesses). DB2 uses two types of threads: the allied threads and the DBATs. Allied threads deal with local (same LPAR) requests, such as those coming from TSO, IMS, CICS, CAF, and RRSAP.

are executed as part of the same enclave's SRBs, and the result set is returned to the application. The enclave is kept alive until the connection goes inactive and becomes a pooled thread (at commit time), or until the thread terminates (in case it cannot become a pooled thread). For information about how enclaves are used by DDF, refer to 7.2.1, "Database Access Threads" on page 271.

In DB2 9, DBM1 and DDF address spaces (as well as IRLM) allocate control blocks above the 2 GB bar and run in 64 bit addressing mode. DDF also uses above-the-bar storage in the z/OS Shared Memory Facility to communicate with DBM1. See Figure 1-12 for an example of the virtual storage layout.

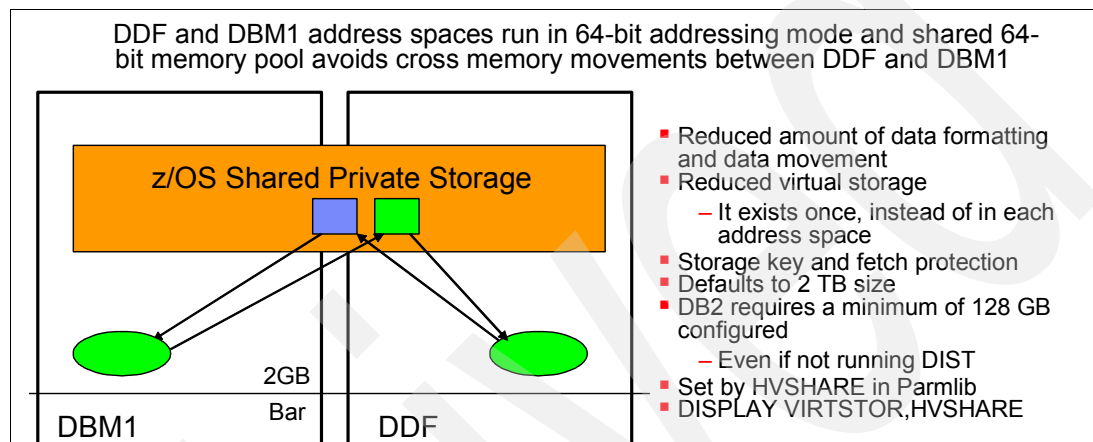


Figure 1-12 DDF - Using shared private storage

DDF has moved almost all of the TCP/IP communication buffers and associated control blocks from ECSA into shared memory, freeing system's resource for other users.

See 3.2.1, "Defining the shared memory object" on page 85 for information about set up and operations.

## 1.6.4 Network protocols used by DDF

DDF can use either SNA or TCP/IP as a network protocol.

### TCP/IP

Figure 1-13 on page 21 shows a sample configuration. In the figure, the AR is on the left, and the AS is on the right. When using DB2 for z/OS as the AR, you have to configure each remote DB2 subsystem that you want to access by specifying the combination of an IP address and a port number. This port number is also called the DRDA SQL port. The information that is necessary to establish the connection to the target subsystem, including IP address, port number, and DB2 location name, is stored in the communication database (CDB) of the local DB2 subsystem. DB2 location name is a unique location name for the accessible server. This is the name by which the remote server is known to local DB2 SQL applications. The location name is the primary entry into the CDB. In addition, DB2 requires that each member of a data sharing group have a resynchronization port number, also known as the resync port, that is unique within the Parallel Sysplex®. In the event of a failure, this unique port number allows a requester to reconnect to the correct member so that units of work that require two-phase commit can be resolved. The resync port only needs to be specified at the AS.

When using TCP/IP as a network protocol, DDF executes TCP/IP services using UNIX System Services/O callable Assembler interface.



Starting with DB2 9 for z/OS, if you do not plan to communicate with remote sites with SNA/APPC and only use TCP/IP, you do not need to define VTAM® to DB2 when you update the BSDS DDF record with an IPNAME value<sup>3</sup>. To use both IPv4 and IPv6 addresses, DB2 requires TCP/IP dual-mode stack support. Dual-mode stack support allows IPv4 address communication with IPv4 partners, and IPv6 address communication with IPv6 partners.

DB2 9 for z/OS is an IPv6 system. All addresses are displayed in IPv6 format, even if only IPv4 addresses are used. This is due to the fact that IP addresses may need to be displayed before DB2 can determine that a dual-mode stack is configured. Hence DB2 chooses to provide consistency by always displaying in IPv6 format.

**Tip:** When using any of the IBM Data Server Drivers as ARs, it is sufficient to provide the IP address, database name, and port of the DB2 for z/OS server either as part of the client application or as a separate configuration or datasource properties file to make a TCP/IP connection. When using DB2 Connect as an AR, you can either use the Client Configuration Assistant to setup a connection to the DB2 for z/OS server or catalog the database through command line processor (CLP).

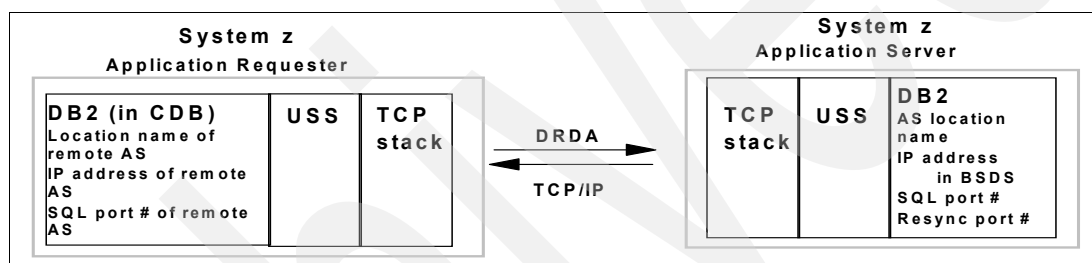


Figure 1-13 Communication using TCP/IP

## SNA

Figure 1-14 on page 22 shows a sample configuration, where DDF uses the Virtual Telecommunication Access Method (VTAM) to communicate with other DB2 for z/OS systems. Communication between subsystems uses a set of Systems Network Architecture (SNA) communication protocols called LU6.2. To talk to a remote DB2 subsystem over SNA, you need to know the LU name and a location name of the remote system. An LU name is the name by which VTAM recognizes the subsystem in the network. The LUNAMES table must be defined in the CDB instead of the IPNAMES name that is used for TCP/IP. Application programs use a location name to indicate which subsystem to access. The local DB2 subsystem also needs to set up its own VTAM ACB to get into the network.

**Restriction:** SNA may be deprecated in a future release of DB2. The usage of TCP/IP is highly recommended. Also note that two-phase commit is no longer supported when you access a DB2 for z/OS server through DB2 Connect Version 8.1 or later release using SNA.

<sup>3</sup> Location, ports and IP address are all recorded in the in BSDS.

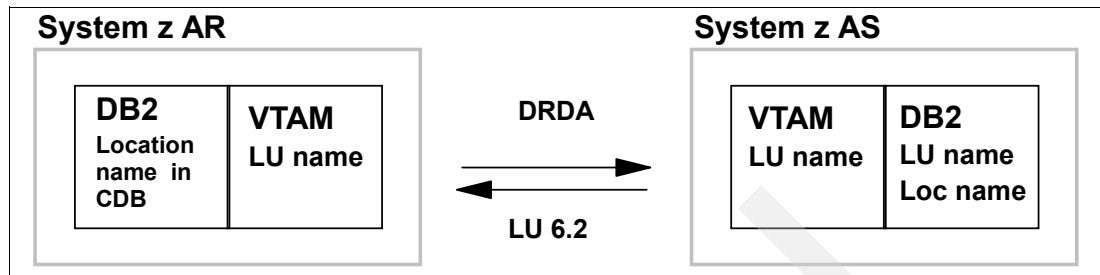


Figure 1-14 Communication using SNA

## 1.7 Connection pooling

Connection pooling is the generic term given to techniques that ensure that the connection resources for distributed database access are pre-loaded, shared, and reused efficiently. There are many different kinds of connection pooling that can be used with DRDA connections to the DB2 for z/OS server.

- ▶ The DB2 for z/OS server provides inactive connection support that provides a mechanism to pool Database Access Threads (DBATs).
- ▶ The IBM Data Server Drivers provide standards-based connection pooling facility that can be exploited by applications using these APIs. They also support connection pooling and transaction pooling (see 1.7.3, “Transaction pooling” on page 25) when sysplex WLB is enabled.
- ▶ DB2 Connect Server provides connection pooling and connection concentration functions.
- ▶ Application Server Environments such as WebSphere provide their own connection pooling software.

As a general rule of thumb, it is beneficial to exploit all the connection pooling facilities that can be applied to any given environment. In general, we recommend concentrator pooling (transaction pooling) over connection pooling. See 1.7.3, “Transaction pooling” on page 25.

### 1.7.1 Inactive connection support in a DB2 for z/OS server

Each inbound connection from distributed environments to the DB2 for z/OS server requires a DDF connection and a DB2 database access thread (DBAT). DB2 for z/OS inactive connection support (formerly referred to as Type 2 inactive thread support or thread pooling) is a mechanism to share few DBATs among many connected applications. See Figure 1-15 on page 23.

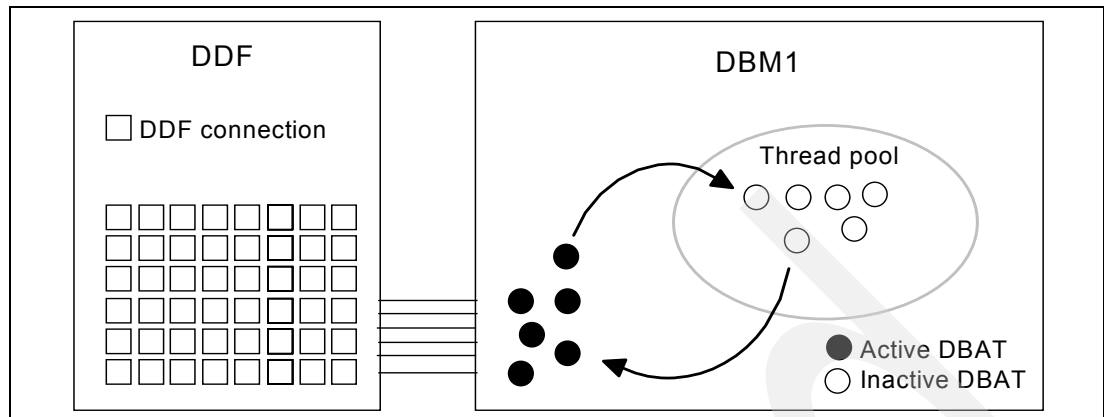


Figure 1-15 Inactive connection support

Inactive connection support operates by separating the distributed connections (in DDF) from the threads (DBATs) that do the work (in DBM1). A pool of DBATs is created for use by inbound DRDA connections. A connection makes temporary use of a DBAT to execute a UOW and releases it back to the pool at commit time, for another connection to use. The result is that a given number of inbound connections require a much smaller number of DBATs to execute the work within DB2. Each DDF connection only consumes approximately 7.5 K of memory inside the DDF address space, whereas each active DBAT consumes approximately 200 K of memory at a minimum, depending on SQL activity. The DSNZPARM CMTSTAT=INACTIVE enables inactive connection support, and starting with DB2 for z/OS V8, this is the default value for the DSNZPARM.

The main advantage of being able to reuse DBATs is related to a greater capacity to support DRDA connections due to the following reasons:

- ▶ CPU savings in DB2, by avoiding repeated creation and destruction of DBATs
- ▶ Real memory savings in z/OS, by reducing the number of DBATs
- ▶ Virtual storage savings in DBM1, by reducing the number of DBATs

See 6.2.2, “Application Servers” on page 249 for examples of deployment of connection reuse and distribution.

It is important to be aware of what prevents a connection from being inactivated:

- ▶ Any WITH HOLD cursors not closed
- ▶ Any declared global temp tables not dropped
- ▶ Any application using packages bound using the KEEP DYNAMIC option (except for certain situations where the requester is able to tolerate the loss of KEEP DYNAMIC environment - See 6.2.2 for details)
- ▶ Held LOB locators

## 1.7.2 Connection pooling using the IBM Data Server Drivers

Connection pooling allows a requester to reuse an existing network connection for a different application once an application disconnects from the connection, either by terminating or by releasing the connection. See Figure 1-16.

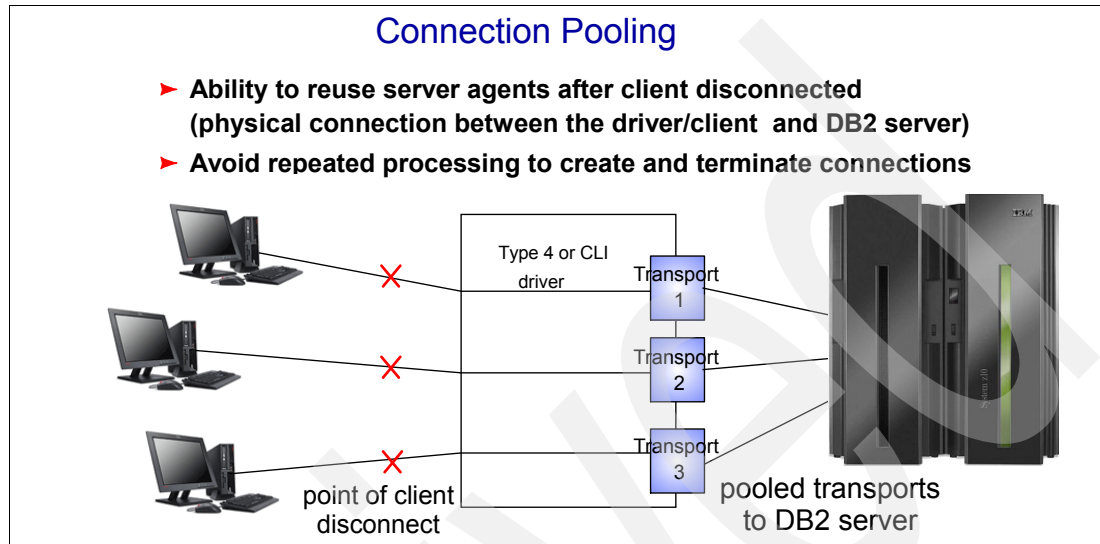


Figure 1-16 Connection pooling

With connection pooling, most application processes do not incur the overhead of creating a new physical connection because the data source can locate and use an existing connection from the pool of connections. When the application terminates, the connection is returned to the connection pool for reuse. After the initial resources are used to produce the connections in the pool, additional overhead is insignificant because the existing connections are reused.

### Benefits of connection pooling

Connection pooling can improve the response time of any application that requires connections, especially Web-based applications. When a user makes a request over the Web to a resource, the resource accesses a data source. Because users connect and disconnect frequently with applications on the Internet, the application requests for data access can surge to considerable volume. Consequently, the total datastore overhead quickly becomes high for Web-based applications, and performance deteriorates. When connection pooling capabilities are used, however, Web applications can realize performance improvements of up to 20 times the normal results.

When using connection pooling, the connection is only available for reuse after the application owning the connection issues a disconnect request. In many client-server applications, users do not disconnect for the duration of the workday. Likewise, most application servers establish database connections at start up and do not release these connections until the application server is shut down. In these environments, connection pooling has little, if any, benefit. However, in Web and client-server environments where the frequency of connections and disconnections is higher, connection pooling will produce significant performance benefits.

### 1.7.3 Transaction pooling

Transaction pooling allows a requester to share a network connection with other applications. It is also referred to as *Sysplex Workload Balancing*. See Figure 1-17.

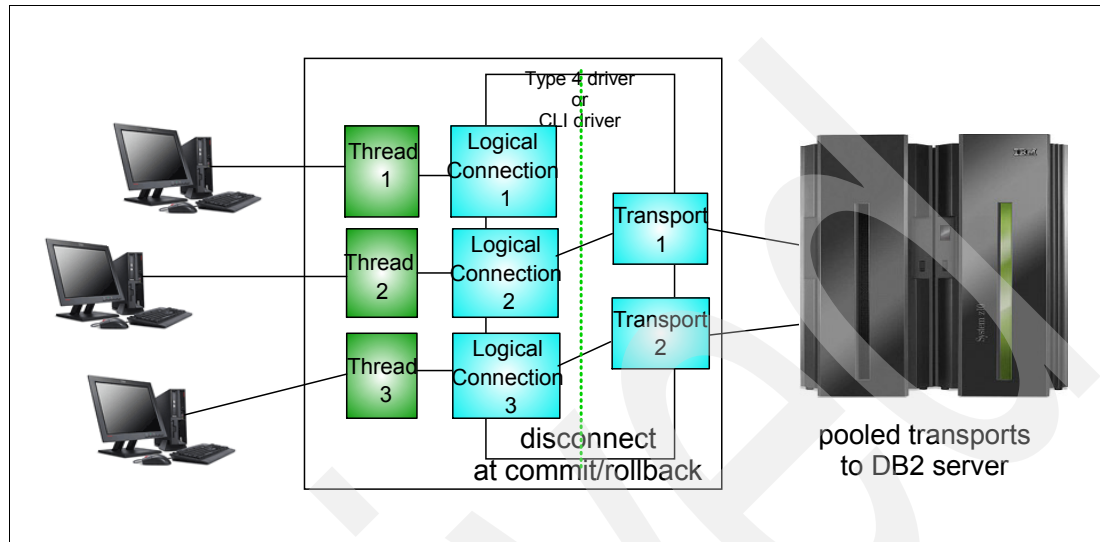


Figure 1-17 Transaction pooling

After a transaction completes, the requester can allow another application to reuse the same physical connection (transport) to the server. During the commit or rollback process, the server indicates whether the connection can be reused by another application. If reuse is allowed, the server returns a group of SQL SET statements to the requester. The requester can then replay these registers to the server, to re-establish the application execution environment prior to the execution of the next transaction for the same application on possibly another connection. The server always returns special registers to client even if the client did not specify the DRDA RLSCONV (release conversation) instance variable.

A sysplex is a collection of DB2 systems (known as members) that form a data sharing group. One or more coupling facilities provide high-speed caching and lock processing for the data-sharing group. The sysplex, together with the WLM dynamic virtual IP address (DVIPA), and the Sysplex Distributor, allow a client to access a DB2 for z/OS database over TCP/IP with network resilience, and distribute transactions for an application in a balanced manner across members within the data-sharing group.

Central to these capabilities is a server list that each member of the DB2 data-sharing group returns on connection boundaries and optionally on transaction boundaries. This list contains the IP address and the workload balancing weight for each DB2 member. With this information, a client can distribute transactions in a balanced manner, or identify the DB2 member to use when there is a communications failure.

The server list is returned on the first successful connection to the DB2 database. Therefore, the initial database connection should be directed at the group DVIPA owned by the Sysplex Distributor. If at least one DB2 member is available, the Sysplex Distributor will route the request to the database. After the client has received the server list, the client directly accesses a DB2 member based on information in the server list.

The IBM Data Server Driver for JDBC and SQLJ is capable of performing sysplex workload balancing functions since Version 8.1 Fixpack 10. Because improvements have been made and seamless client reroute has been added, V9 FixPack 5 and V9.5 FixPack 1 are the

recommended minimum levels. Starting with Version 9.5 FixPack 3, IBM Data Server Clients and non-Java-based data server drivers that have a DB2 Connect license, can also access a DB2 for z/OS sysplex directly. Licensed clients no longer need to go through a middle-tier DB2 Connect (gateway) server to use sysplex capabilities.

With workload balancing (see Figure 1-18), DB2 for z/OS and WLM ensure that work is distributed efficiently among members of the data sharing group and that work is transferred to another member of a data sharing group if one member has a failure.

When workload balancing is enabled, the driver gets frequent status information about the members of a data sharing group. The driver uses this information to determine the data sharing member to which the next transaction should be routed.

The IBM Data Server Driver for JDBC and SQLJ uses transport objects and a global transport objects pool to support the workload balancing by performing both transaction pooling and connection concentration. There is one transport object for each physical connection to the data source. When you enable the connection concentrator and workload balancing, you set the maximum number of physical connections to the data source at any point in time by setting the maximum number of transport objects.

To configure non-Java-based client sysplex support, specify settings in the db2dsdriver.cfg configuration file, explained in “.NET Provider/CLI Driver” on page 253. The DB2 for z/OS requester uses the server list for workload balancing on a connection boundary, but it cannot balance workload on a transaction boundary.

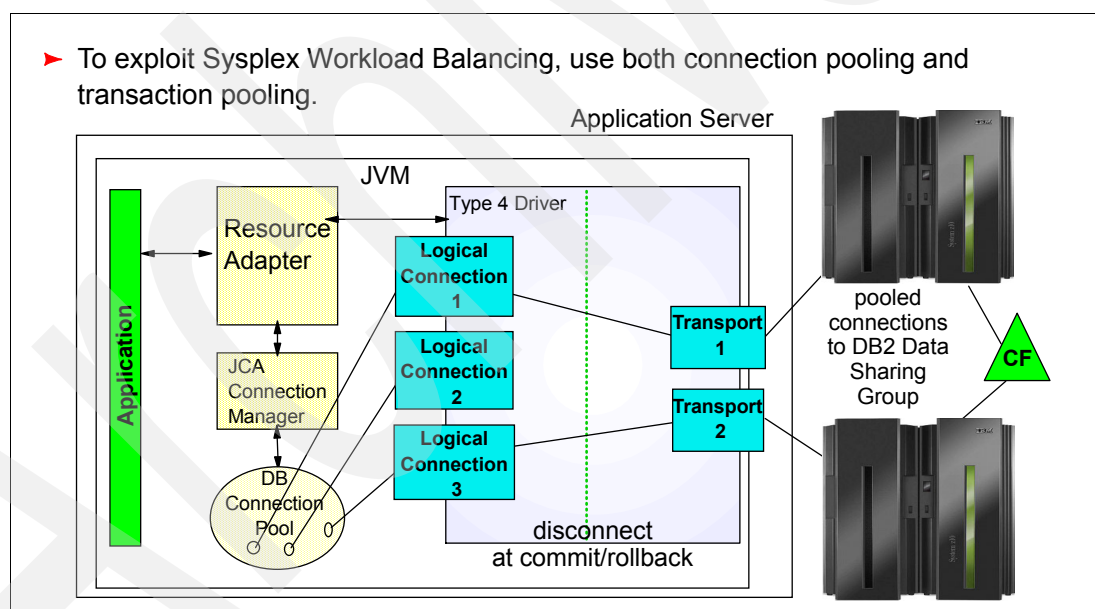


Figure 1-18 Sysplex workload balancing

## 1.8 Federated data support

Today's business climate demands fast analysis of large amounts and disparate sources of business critical data. Businesses must access not only traditional application sources such as relational databases, but also XML documents, text documents, scanned images, video clips, news feeds, Web content, e-mail, analytical cubes, and special-purpose stores. Data federation is a way of providing an end-to-end solution for transparently managing the diversity of data. Data federation can quickly build the framework for such a solution by

allowing applications to meet the need to adapt to business change, while maintaining a business infrastructure that keeps up with the demands of the marketplace.

DB2's federated database functionalities extend the reach of all units of work letting you access data in multiple sources in one SQL statement.

IBM developed DB2 Data Joiner several years ago as the first vehicle for federated technology. Later federated database technology was delivered with DB2 UDB Version 7.1. This product provided a unified access (information integration) to diverse and distributed data belonging to the IBM DB2 family as well as Informix IDS. IBM DB2 Information Integrator V8.1 extended the federated approach by providing the ability to synchronize distributed data without requiring that it be moved to a central repository. For details, see *Data Federation with IBM DB2 Information Integrator V8.1*, SG24-7052, and *Publishing IMS and DB2 Data using WebSphere Information Integrator: Configuration and Monitoring Guide*, SG24-7132.

The current products, branded as of September 30th, 2008<sup>4</sup> as members of the IBM InfoSphere family, are shown in Figure 1-19.

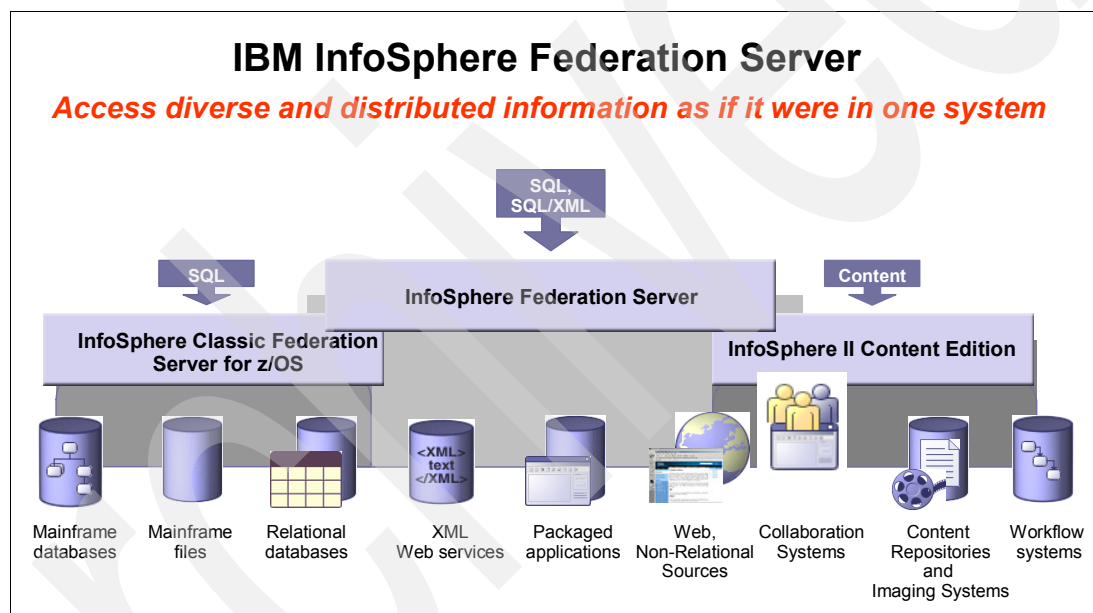


Figure 1-19 InfoSphere Federation Server products

The InfoSphere brand represents integration and openness, which are integral to the Federation Server software portfolio and its role in providing organizations with real-time, integrated access to disparate and distributed information across IBM and non-IBM sources. Find more information at the following Web page:

<http://www.ibm.com/software/data/integration/>

In this section we briefly introduce the relevant Federation Server products.

<sup>4</sup> IBM United States Software Announcement 208-159, dated September 30, 2008: IBM InfoSphere Information Server offers globalization, native AIX® and Solaris™ 64-bit support, enhanced deployment, and improved SOA capabilities

## 1.8.1 IBM InfoSphere Federation Server

IBM InfoSphere Information Server achieves new levels of information integration and flexibility by delivering industry-leading capabilities for understanding, cleansing, transforming, and delivering information. Figure 1-20 on page 29 summarizes the functional areas.

The recently enhanced areas are as follows:

- ▶ **Enhanced deployment**  
Ensures secure deployment and simplified metadata integration.
- ▶ **Connectivity to industry-leading applications, databases, and file systems**  
IBM InfoSphere Information Server is designed for automated and optimized access to data stored behind industry-leading applications, databases, and file systems without the need for extensive and complex code customization.
- ▶ **Improved information as a service**  
Extends the service-oriented reach of IBM InfoSphere Information Server to InfoSphere Master Data Management Server, IBM InfoSphere Classic Federation Server, and Oracle®.
- ▶ **Direct data lineage integration capabilities**  
Extends IBM Metadata Workbench analysis from within Cognos® Framework Manager 8.4.
- ▶ **Data quality enhancements**  
Along with globalization and enhanced deployment capabilities, InfoSphere QualityStage has been enhanced with significant match improvements.

Databases include the DB2 product family, Oracle, Microsoft® SQL Server®, and Sybase.



## InfoSphere Federation Server

- **Transparent**
  - Appears to be one source
  - Independent of how and where data is stored
  - Applications continue to work despite of any change in how data is stored
- **Heterogeneous**
  - Accesses data from diverse sources
  - Relational, Structured, XML, messages, Web content ...
- **Extensible**
  - Bring together almost any data source.
  - Wrapper Development Toolkit
- **High Function**
  - Full query support against all data
  - Capabilities of sources as well
- **Autonomous**
  - Non-disruptive to data sources, existing applications, systems.
- **High Performance**
  - Optimization of distributed queries

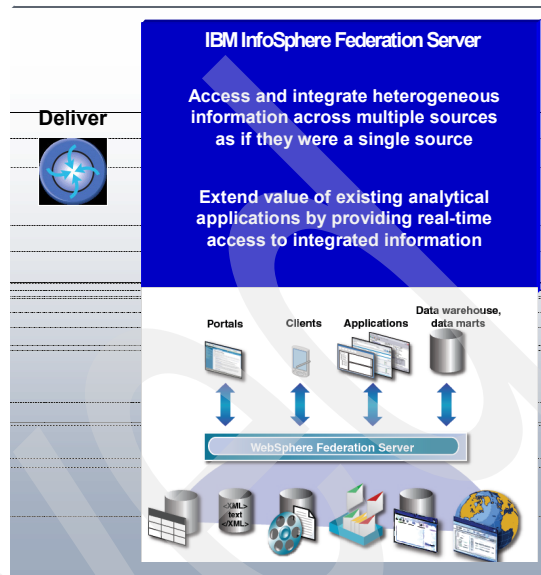


Figure 1-20 InfoSphere Federation Server

### 1.8.2 IBM InfoSphere Classic Federation Server for z/OS

IBM InfoSphere Classic Federation Server for z/OS is the current product that provides direct, real-time SQL access to mainframe databases and files without mainframe programming, as shown in Figure 1-21. It maps logical relational table structures to existing physical mainframe databases and files. UNIX, Windows, and Linux tools and applications issue standard SQL commands to these logical tables.

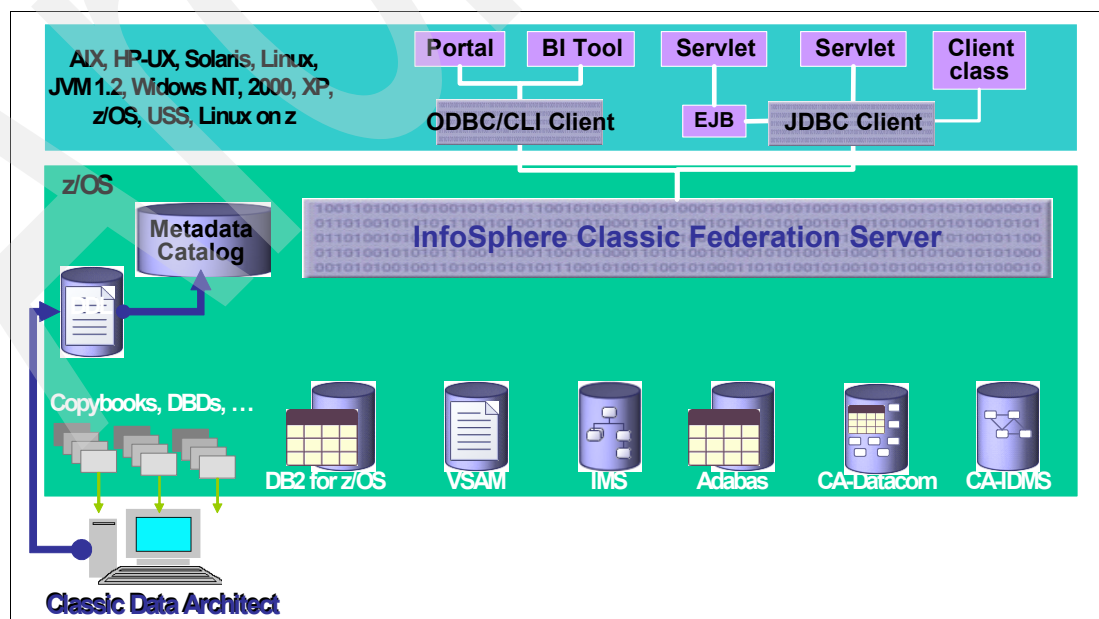


Figure 1-21 Classic Federation Server for z/OS

InfoSphere Classic Federation Server for z/OS dynamically generates native data access commands that are optimized for each database and file type. Results are automatically translated and reformatted into relational rows and columns providing seamless integration of all mainframe data assets without proprietary programming.

InfoSphere Classic Federation Server for z/OS can also extend IBM InfoSphere Federation Server's access to non-relational mainframe data sources. InfoSphere Classic Federation Server lets applications access diverse and distributed data (including multivendor mainframe and distributed, structured and unstructured, public and private) as though it were a single database.

Classic Federation Server V9.5 provides extensions to the Classic Data Architect to help automate the creation of metadata for more users and extend visual metadata management to more complex scenarios. This improves the usability of the tool while also enabling more rapid metadata creation and management. In addition, Classic Federation Server for z/OS V9.5 provides an interactive view of the actual settings as well as a means for temporarily and permanently changing configuration settings, greatly simplifying the management of the operational platform and performance optimization processes.

Classic Federation Server for z/OS V9.5 supports new 64-bit clients and JDBC 3.0 SQL.

IBM InfoSphere Classic Federation Server for z/OS V9.5 can access data stored in VSAM, IMS, CA-IDMS, CA-Datcom, Software AG Adabas, and DB2 for z/OS databases by dynamically translating JDBC and ODBC SQL statements into native read/write APIs.

Data client software requirements depend on the data sources that are accessed. The client software must be acquired separately unless specified otherwise. The client software must be installed on the same system as the InfoSphere Federation or Replication Server.

The relationships between products and their respective functionalities are shown in Figure 1-22.

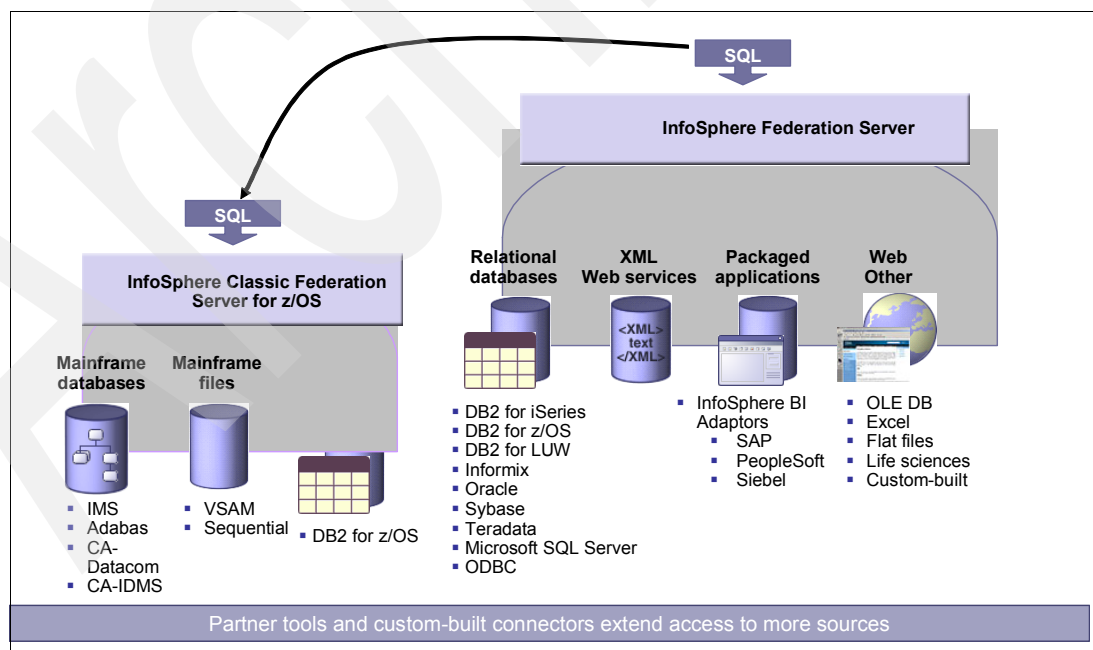


Figure 1-22 Supported data sources

## **Federation functions**

- ▶ Federated access in InfoSphere Federation Server supports all the data sources shown in Figure 1-22 on page 30.
- ▶ Federated access in the DB2 homogeneous federation feature supports access to DB2 for Linux, UNIX, and Windows, DB2 for z/OS, DB2 for System i®, and Informix.

## **Replication and Event Publishing**

- ▶ SQL-based replication supports:
  - DB2, Informix Dynamic Server, Microsoft SQL Server, Oracle, and Sybase Adaptive Server Enterprise as sources and targets.
  - Informix Extended Parallel Server and Teradata as targets.
  - DB2 for z/OS support requires InfoSphere Replication Server for z/OS.
  - DB2 for iSeries® support requires IBM DB2 DataPropagator for System i.
- ▶ Queue-based replication supports:
  - DB2 for Linux, UNIX, and Windows and DB2 for z/OS as sources and targets.
  - Informix Dynamic Server, Microsoft SQL Server, Oracle, and Sybase Adaptive Server Enterprise as targets.
  - DB2 for z/OS support requires InfoSphere Replication Server for z/OS.
- ▶ Data event publishing supports:
  - DB2 for Linux, UNIX, and Windows and DB2 for z/OS.
  - DB2 for z/OS support requires InfoSphere Data Event Publisher for z/OS.

Archived

## Distributed database configurations

In this chapter we discuss different scenarios for implementing DB2 for z/OS in a distributed database business environment. We show some of the most commonly used configurations. We mention protocols, installation and setup considerations, and application programming interfaces that you can use. Implementations using DB2 Private Protocol are not discussed in this chapter, as DB2 Private Protocol function has been deprecated. We do not discuss Systems Network Architecture (SNA) support in this chapter, as SNA support for distributed access has not been enhanced.

As mentioned in 1.4.1, “DB2 for z/OS” on page 14, DB2 for z/OS implements both the DRDA AR and DRDA AS functions.

DB2 for Linux, UNIX, and Windows (DB2 for LUW) provides DRDA AS functions as a standard feature. DB2 for LUW is delivered to the market in a set of Editions. The difference between the editions is as follows:

- ▶ **DB2 Express Edition**  
DB2 Express Edition can be run on servers with a maximum of 2 CPUs and 4 GB of real memory.
- ▶ **DB2 Workgroup Server Edition**  
DB2 Workgroup Server Edition (WSE) can be run on servers with a maximum of 4 CPUs and 16 GB of real memory.
- ▶ **DB2 Enterprise Server Edition**  
DB2 Enterprise Server Edition (ESE) can run on any size server with any amount of real memory

Among the DB2 for LUW products, only DB2 for LUW Enterprise Server Edition (DB2 for LUW ESE) includes the DRDA AR functions. DB2 Connect provides the DRDA AR functions that allow the other DB2 for LUW products to access DB2 for z/OS data.

Applications that do not require a local DB2 for LUW database can access DB2 for z/OS data through DB2 Connect or by acquiring a DB2 Connect license and using the DRDA AR functions in one of the following products:

- ▶ IBM Data Server Driver for JDBC and SQLJ
- ▶ IBM Data Server Driver for ODBC and CLI
- ▶ IBM Data Server Driver Package
- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Client

We discuss configurations using DB2 for z/OS, either as a requester or a server, or both. We also discuss configurations for two-tier and three-tier access to DB2 for z/OS, as a server, from DB2 for LUW, DB2 Connect and the IBM Data Server products listed above. Throughout this chapter we focus on requesters and servers, rather than AR and AS functions. This is because DRDA has evolved to include application requester (AR), application server (AS) and database server (DS) functions, and DB2 for z/OS, as well as other DRDA participants, may perform different DRDA functions at different times or in different scenarios.

This chapter consists of the following sections:

- ▶ “DB2 for z/OS both as a requester and a server” on page 35
- ▶ “DB2 for LUW and DB2 for z/OS” on page 37
- ▶ “IBM Data Server Drivers and Clients as requesters” on page 40
- ▶ “DB2 Connect to DB2 for z/OS: Past, present, and future” on page 51
- ▶ “DB2 Connect Server on Linux on IBM System z” on page 58
- ▶ “DB2 for z/OS requester: Any (DB2) DRDA server” on page 63
- ▶ “XA Support in DB2 for z/OS” on page 63

## 2.1 DB2 for z/OS both as a requester and a server

In this section, we discuss configurations where both the requester and the server are DB2 for z/OS subsystems.

### 2.1.1 Basic configuration

Figure 2-1 shows the DB2 for z/OS configuration. A group of address spaces, indicated as DB2 in the figure, provides database functions and is called a DB2 subsystem. A DB2 subsystem includes the DIST, DBM1, MSTR, and IRLM address spaces, representing the Distributed Data Facility (DDF), database services, system services, and locking manager address spaces, respectively.

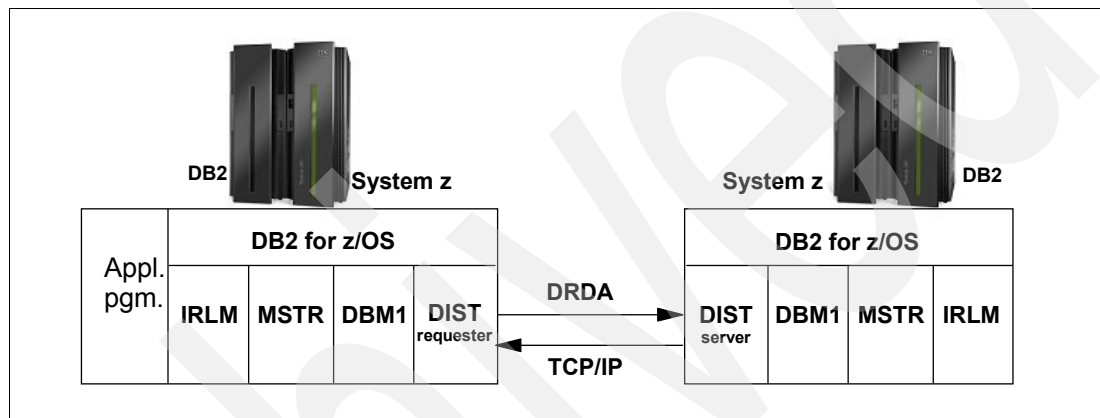


Figure 2-1 Connecting from DB2 for z/OS to DB2 for z/OS

To send requests to or receive requests from remote subsystems, you have to start the DDF address space. You can stop or start DDF independently from other DB2 address spaces. However, the DDF address space (ssidDIST) itself is always started together with the other DB2 address spaces when the DDF DSNZPARM is set to COMMAND or AUTO. By stopping and starting DDF, you only affect the communication with other systems. DDF can act as a requester, an intermediate server for a remote requester, and as a server. In Figure 2-1, an application program attached to the DB2 subsystem on the left requests data from the DB2 subsystem on the right.

#### Configuration overview

TCP/IP is the network protocol. DB2 stores connectivity information in the communication database (CDB) and the bootstrap data set (BSDS). An application uses the target DB2 LOCATION name to address a DB2 subsystem.

The following TCP/IP information is necessary to identify DB2 for z/OS server subsystems:

- ▶ IP address or domain name of the system where the DB2 subsystem resides.
- ▶ Port numbers or service names to specify port numbers. You have to define a server port (also DRDA SQL port) and a resynchronization port<sup>1</sup>.

<sup>1</sup> In the event of a failure, this unique port number allows a requester to reconnect to the subsystem or member so that units of work that require two-phase commit can be resolved. Remote DRDA partners record DB2's resynchronization port number in their recovery logs.

At the DB2 for z/OS requester in a TCP/IP environment, the combination of IP address and port number specifies the target subsystem in the network, which is linked with a LOCATION name in the CDB.

Application programs can access a remote subsystem using the same APIs as are used for the local subsystem. Both static and dynamic SQL are available.

## 2.1.2 Parallel sysplex environment

Parallel sysplex technology is a solution to provide highest levels of availability and scalability to mission-critical systems. DB2 for z/OS can run in a parallel sysplex environment (Figure 2-2).

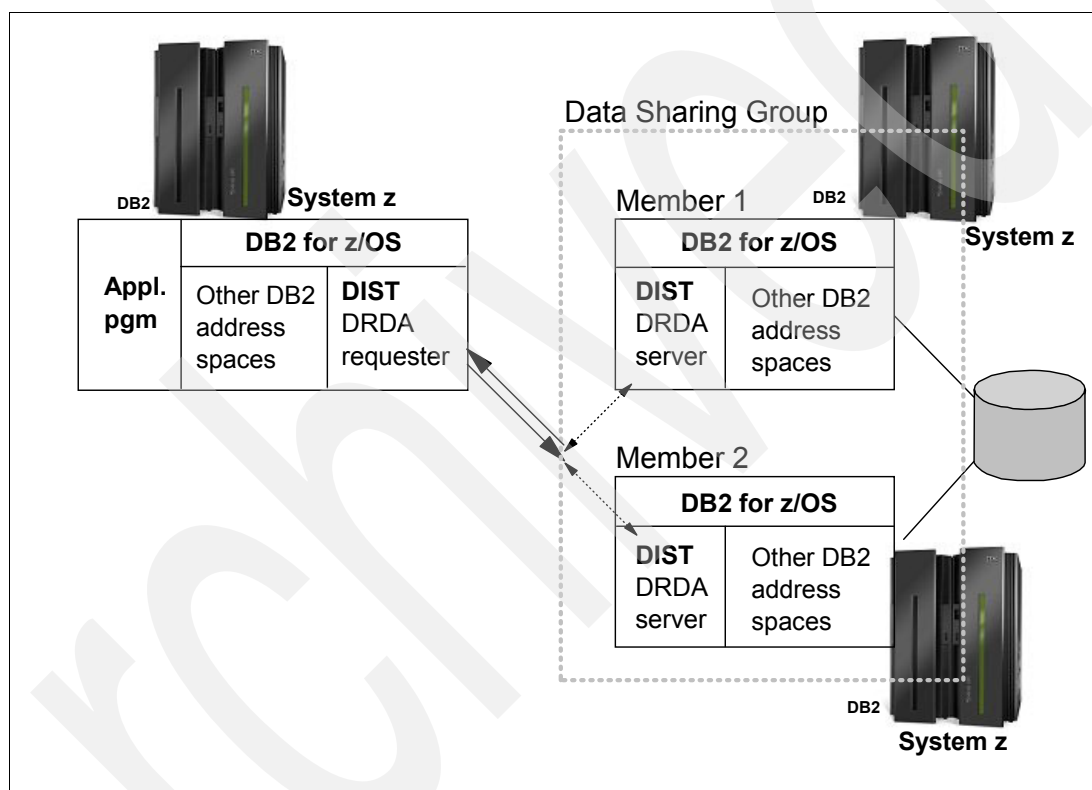


Figure 2-2 Connecting to a DB2 data sharing group from a DB2 for z/OS system

In this configuration, a set of DB2 subsystems (called a data sharing group) shares a set of system and user tables. Each DB2 subsystem that is part of the data sharing group is called a member. A data sharing group can support more threads than a single DB2 subsystem. The maximum number of DDF threads or connections in a data sharing group is the sum of the number of threads or connections in each of the DB2 members in that data sharing group.

It is worth noting that DB2 for LUW Enterprise Server Edition (ESE) requesters (see 2.2, “DB2 for LUW and DB2 for z/OS” on page 37), IBM Data Server Drivers and Clients (see 2.3, “IBM Data Server Drivers and Clients as requesters” on page 40), and DB2 Connect requesters and servers (see 2.4, “DB2 Connect to DB2 for z/OS: Past, present, and future” on page 51), can also connect to a data sharing group.



The data sharing group has a single-system image for requesting applications. Requesting applications use the LOCATION NAME of the data sharing group to direct their SQL requests to that group. There is a single location name that identifies the data sharing group. DB2 for z/OS also supports LOCATION ALIAS to allow requesting applications to specify a subset of the members of the data sharing group.

For more information about how to connect to a data sharing group, see Chapter 6, “Data sharing” on page 233.

## 2.2 DB2 for LUW and DB2 for z/OS

DB2 for LUW ESE provides support for local DB2 databases as well as DRDA server and requester functions. Applications local to DB2 for LUW need no additional products to access local DB2 data or data from other DRDA servers. Remote applications, including applications on a System z with DB2 for z/OS, can access DB2 data stored in DB2 for LUW.

### 2.2.1 DB2 for LUW ESE as requester to DB2 for z/OS server

If you have an application that currently uses data stored in DB2 for LUW ESE, that application can also access data stored in DB2 for z/OS, even in the same transaction. You can also deploy applications on this Windows, UNIX, or Linux system that only access data stored in DB2 for z/OS. Figure 2-3 shows DB2 for LUW as an application requester accessing data from DB2 for z/OS.

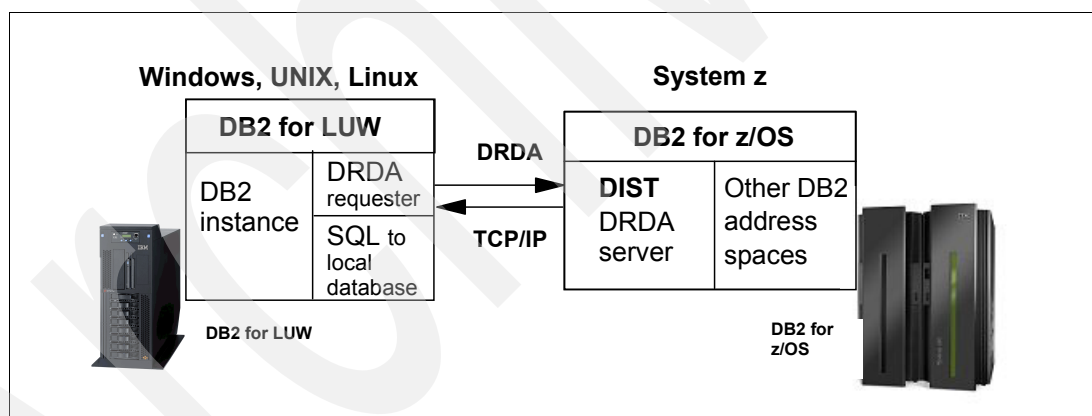


Figure 2-3 DB2 for LUW ESE requester connecting to DB2 for z/OS.

If the application also accesses a local DB2 for LUW database, DB2 for LUW establishes two processes: one for DRDA to DB2 for z/OS, the other for the local database access.

### 2.2.2 DB2 for z/OS as requester to DB2 for LUW as server

DB2 for LUW provides DRDA server functionality as a standard function. Applications attached to DB2 for z/OS can access data stored in DB2 for LUW, with a network connection, without additional software. DB2 for z/OS requires an entry in its Communication Database (CDB) to perform DRDA requester functions. Figure 2-4 on page 38 shows DB2 for z/OS using DDF to access data in a DB2 for LUW database. As in Figure 2-3, an application attached to DB2 for z/OS can access data stored in DB2 for z/OS and data stored in DB2 for LUW in the same transaction. In this case the server could be DB2 for LUW ESE or DB2 for LUW WSE, because the DRDA requester functions are not required.

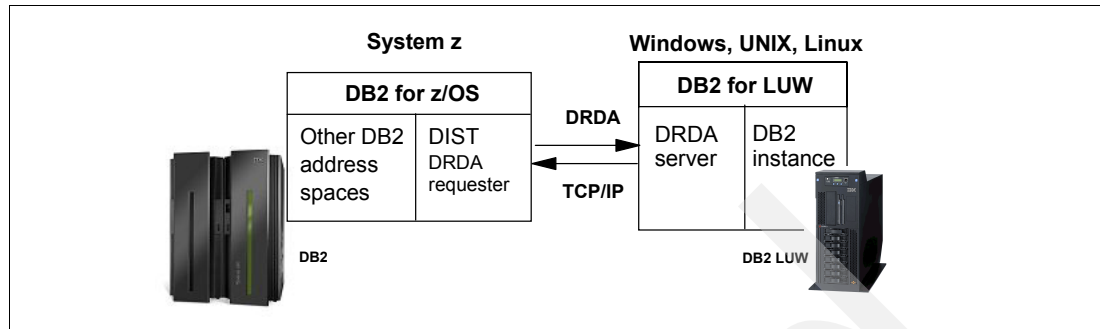


Figure 2-4 DB2 for z/OS requester connecting to DB2 for LUW ESE or DB2 for LUW WSE

One difference from the previous example is that for a given application DB2 for z/OS uses a single process to provide access to both DB2 for LUW data and DB2 for z/OS data.

### 2.2.3 DB2 for z/OS as an intermediate server

DDF can perform as a gateway by acting as both server and requester for the same connection. In this case we use the terms ‘upstream requester’, ‘intermediate server’ and ‘downstream server’. Figure 2-5 illustrates an upstream requester accessing data in DB2 for z/OS, then accessing data on a downstream server. DB2A is the intermediate server when the application accesses data on the downstream server.

In this example, the requester application issues a `CONNECT TO DB2A` and a `SELECT FROM TABLE1`. DB2A’s DDF serves as a DRDA AS to satisfy those statements. Next, the application issues a `SELECT FROM DB2B.MYDB.TABLE2`. In this case, DB2A’s DDF acts as an AS on behalf of the requesting application to resolve the three-part name and direct the request to DB2B, which provides the database server (DS) function.

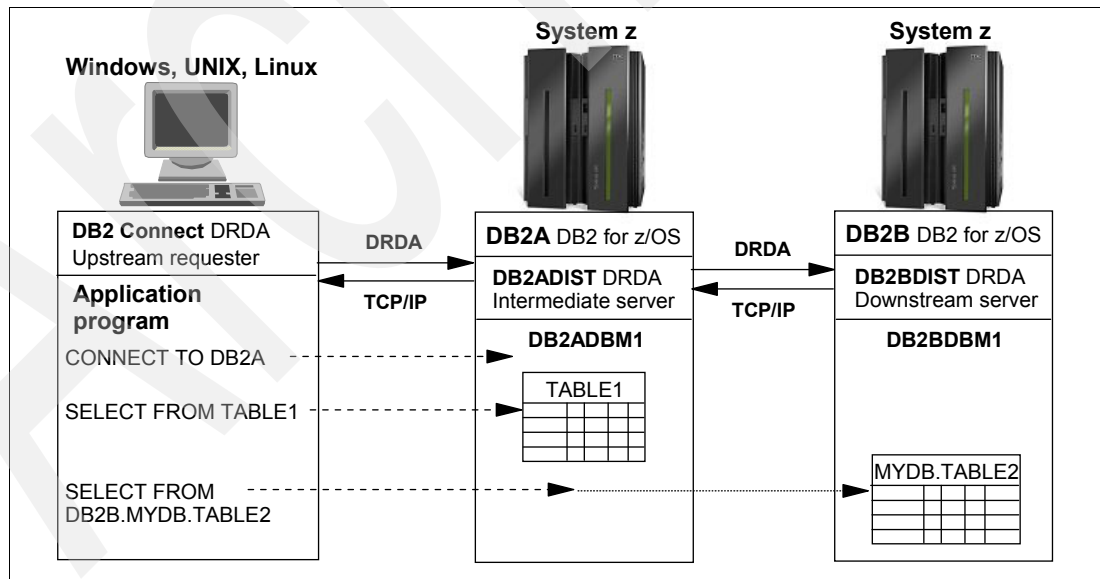


Figure 2-5 DB2A as an intermediate server between a requester and a server

The same concept applies to any DRDA-supporting product acting as a downstream server. DB2 for z/OS can be an intermediate server from any DRDA upstream requester to any DRDA downstream server.

## 2.2.4 DB2 for z/OS as requester to a federation server

There is a special case where DB2 for z/OS acts as a DRDA requester to a federation server on behalf of locally attached applications. In this case, DB2 for z/OS is not the two-phase commit (2PC) coordinator. If all the data sources accessed during the unit of work (UOW) support 2PC, for example in a global XA transaction, there are no issues. But if one of the data sources does not support 2PC and is updated as part of one-phase commit (1PC) processing, then there are special DRDA flows between DB2 for z/OS and the federation server to determine how to handle updates to the 1PC data source in the UOW.

This support was added to DB2 for z/OS as part of DB2 9 for z/OS, and was retrofitted to DB2 for z/OS V8. Refer to Figure 2-6 as we discuss this situation. In this case DB2A acts as a DRDA requester on behalf of an IMS or CICS transaction. IMS and CICS are non-DRDA 2PC coordinators with respect to DB2, which is a 2PC participant. In Figure 2-6, an application attempts to update several data sources, including two through a federation server (DB2C, a DB2 for LUW system). TABLE1 is a DB2 for z/OS table at a remote site, in DB2B. TABLE2 is a DB2 for LUW table in a database local to the federation server, DB2C. TABLE3 is a non-DRDA data source, such as a file, a non-relational database or non-DRDA relational database, that is associated to the federation server. TABLE4 is a DB2 for z/OS table in the local DB2A subsystem.

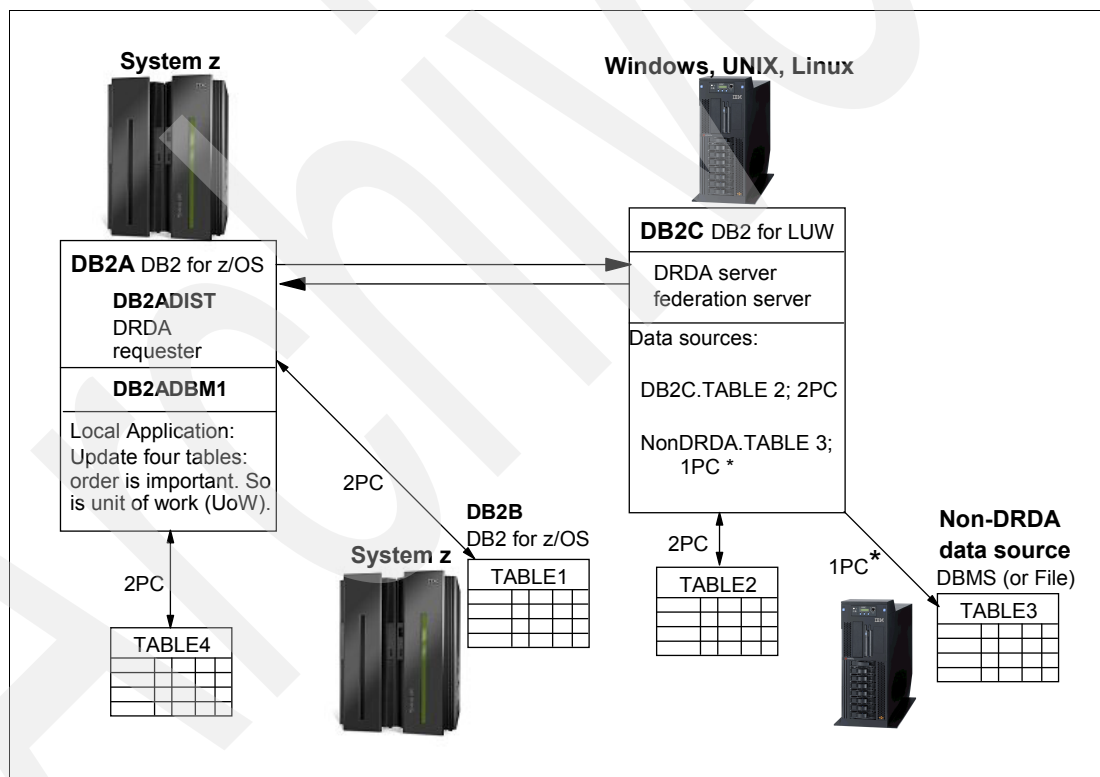


Figure 2-6 DB2 for z/OS as requester in a federation server, unprotected update scenario

The business requirement in our scenario is that the application update all four tables. The requirement to update all four tables presents a distinct challenge, because the non-DRDA data source only supports one-phase commit (1PC). Therefore, any update to TABLE3 must be an unprotected update. That is, it cannot be coordinated with other updates to tables where the database supports 2PC. We will describe several options the application may pursue.

First, the application may try to update the tables in order in a single UOW. In summary, the application would look similar to Example 2-1.

*Example 2-1 Updating tables in order in a single UOW*

---

```
UPDATE TABLE1;  
UPDATE TABLE2;  
UPDATE TABLE3;  
UPDATE TABLE4;  
COMMIT;
```

---

In this case, the first two updates would be in process when the application attempted to update TABLE3. Because the update to TABLE3 is an unprotected update, and other 2PC updates were already in progress, the update to TABLE3 would fail and the whole transaction would roll back.

Second, the application may try to update TABLE3 first, then update the other tables. In this case, the update to TABLE3 would succeed, and the federation server would respond back to DB2A that an unprotected update had been performed. At this point, DB2A would disallow updates to other sources. Any access other than SELECT for TABLE1, TABLE2, or TABLE4 would result in the application receiving a -919 SQL code.

Third, the application might try to update all the other tables first and to update TABLE3 last. However, just as in the first scenario, the fact that other 2PC updates had already occurred would cause DB2A to disallow the unprotected update in this UOW.

Finally, the application would have to approach the 2PC data sources separately from the 1PC data source. That is, the application would either have to update TABLE3, commit, and perform the other updates dependent upon the result of the first, or update TABLE1, TABLE2, and TABLE4, issue a commit, and then update TABLE3.

The point to remember is, when unprotected updates are allowed, other updates in the same UOW are disallowed.

For more information about federation servers, refer to 1.8, “Federated data support” on page 26.

## 2.3 IBM Data Server Drivers and Clients as requesters

The IBM strategy is to remove the reliance on the DB2 Connect modules and replace DB2 Connect with the IBM Data Server Drivers or Clients. While DB2 Connect licenses (in the form of DB2 Connect license files) are still required, you can replace DB2 Connect modules with the IBM Data Server Drivers or Clients and receive equivalent or superior function. In addition, you can reduce complexity, improve performance, and deploy application solutions with smaller footprints for your business users.

With DB2 for LUW Version 9.5 FixPack 3 or FixPack 4 you can implement the DRDA AR functions for your distributed applications with varied degrees of granularity. Instead of the current function and large footprint of DB2 Connect, you can choose from the IBM Data Server Drivers, the IBM Data Server Runtime Client, and the IBM Data Server Client. The IBM Data Server Drivers include:

- ▶ IBM Data Server Driver for JDBC and SQLJ
- ▶ IBM Data Server Driver for ODBC and CLI
- ▶ IBM Data Server Driver Package

In this section we introduce these drivers and clients and discuss situations where you can take advantage of them to reduce complexity and improve performance and availability for distributed access to DB2 for z/OS data.

### 2.3.1 DB2 distributed clients: Historical view

IBM has delivered a variety of client products for applications, application developers, and database administrators to support distributed access to data stored in DB2 for z/OS. The names and function of these products in recent DB2 for LUW versions are highlighted in Table 2-1.

*Table 2-1 Recent history of DB2 client products*

DB2 8.2 Clients	DB2 9 Clients	DB2 9.5 and 9.7 Clients
DB2 Administration Client  DB2 Application Development Client	DB2 Client	IBM Data Server Client
DB2 Run-Time Client	DB2 Runtime Client	IBM Data Server Runtime Client
Java Common Client	IBM DB2 Driver for JDBC and SQLJ	IBM Data Server Driver for JDBC and SQLJ
n/a	IBM DB2 Driver for ODBC and CLI	IBM Data Server Driver for ODBC and CLI
n/a	n/a	IBM Data Server Driver Package

Each of the DB2 9.5 clients in the right-most column is described in the sections that follow.

### 2.3.2 IBM Data Server Drivers and Clients overview

The IBM Data Server products include DRDA AR functions in FixPack 3. With DB2 9.5 FixPack 3, installing DB2 Connect modules is no longer required to connect to mid-range or mainframe databases, although a license for DB2 Connect is still required. You may choose to use DB2 Connect as a server in some circumstances. This is discussed in detail in 2.5, “DB2 Connect Server on Linux on IBM System z” on page 58.

In this section we briefly describe these drivers and clients, including examples showing distributed applications accessing data stored in DB2 for z/OS without using a DB2 Connect Server. Following these examples we include a table that compares the driver and client products. Start with the IBM DB2 9.5 Information Center at the following Web page for more information about these products:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.swg.im.dbclient.install.doc/doc/c0022612.html>

#### IBM Data Server Driver for JDBC and SQLJ

This driver is for applications using only Java. This driver provides support for client applications and applets that are written in Java using JDBC or SQLJ. This is the latest driver from IBM to support JDBC connectivity. We present a brief introduction to JDBC drivers, then return to the other IBM Data Server products.

## DB2 support for JDBC drivers

The DB2 product includes support for two types of JDBC driver architecture, the Type 2 driver and the Type 4 driver.

► Driver for JDBC Type 2 connectivity (Type 2 driver)

Type 2 drivers are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Because of the native code, their portability is limited. Use of the Type 2 driver to connect to DB2 for z/OS is recommended for WebSphere Application Server running on System z.

► Driver for JDBC Type 4 connectivity (Type 4 driver)

Type 4 drivers are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source. The JDBC Type 4 driver is recommended to connect distributed Java applications to DB2 for z/OS data.

IBM ships two versions of the JDBC Type 4 driver with the IBM Data Server Driver for JDBC and SQLJ V9.5 FP3 product:

- Version 3.5x is JDBC 3.0-compliant. It is packaged as db2jcc.jar and sqlj.zip and provides JDBC 3.0 and earlier support.
- Version 4.x is JDBC 3.0-compliant and supports some JDBC 4.0 functions. It is packaged as db2jcc4.jar and sqlj4.zip.

The Type 4 driver provides support for distributed transaction management. This support implements the Java 2 Platform, Enterprise Edition (J2EE™), Java Transaction Service (JTS), and Java Transaction API (JTA) specifications, which conform to the X/Open standard for distributed transactions (Distributed Transaction Processing: The XA Specification, available from the following Web page:

<http://www.opengroup.org>

We include an example of XA support in 2.7, “XA Support in DB2 for z/OS” on page 63.

Applications that use the IBM Data Server Driver for JDBC and SQLJ to access DB2 for z/OS data across a network implement the Type 4 driver. In Figure 2-7 an application uses the IBM Data Server Driver for JDBC and SQLJ to access a standalone DB2.

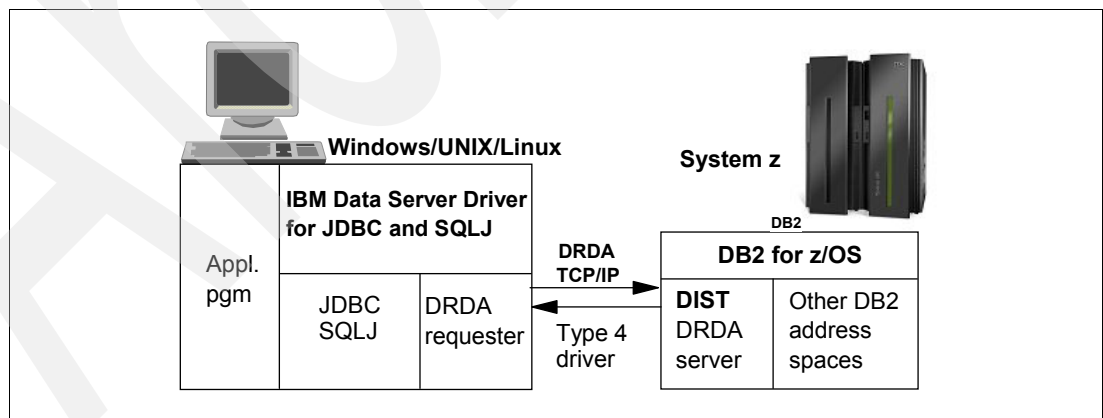


Figure 2-7 IBM Data Server Driver for JDBC and SQLJ connecting directly to DB2 for z/OS

In the remainder of the discussion and examples of the IBM Data Server Driver for JDBC and SQLJ, we refer to the Type 4 driver support.

## IBM Data Server Driver for ODBC and CLI (CLI driver)

This product is for applications using ODBC or CLI only and provides a lightweight deployment solution designed for ISV deployments. This driver, also referred to as *CLI driver*, provides runtime support for applications using the ODBC API or CLI API, without the need of installing the IBM Data Server Client or the IBM Data Server Runtime Client.

The CLI driver is conceptually similar to the JDBC Type 4 driver. The CLI driver is packaged in a small footprint, providing the DRDA AR functions necessary to connect to DB2 for z/OS for those application scenarios where you do not require robust tools, development or administration functions.

In Figure 2-8, an application uses the CLI driver to access DB2 for z/OS data directly.

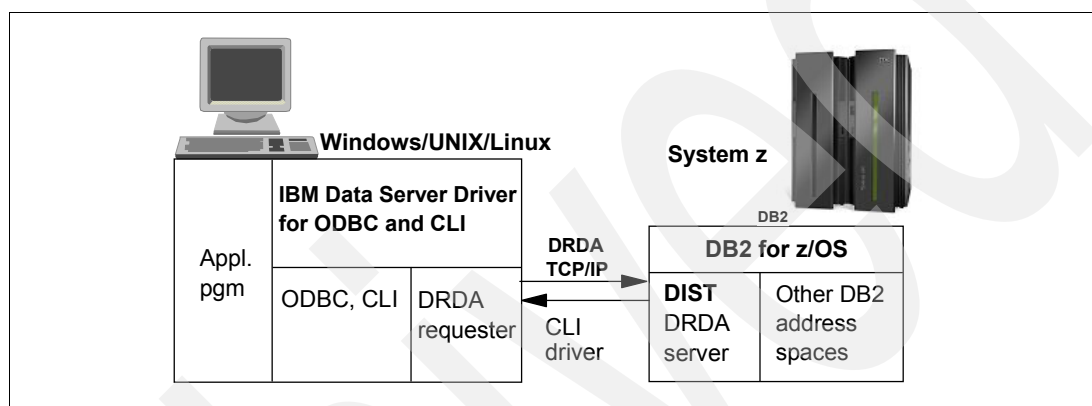


Figure 2-8 IBM Data Server Driver for ODBC and CLI connecting directly to DB2 for z/OS

## IBM Data Server Driver Package

IBM Data Server Driver Package provides a lightweight deployment solution providing runtime support for applications using ODBC, CLI, .NET, OLE DB, open source, or Java APIs without the need of installing Data Server Runtime Client or Data Server Client. This driver has a small footprint and is designed to be redistributed by independent software vendors (ISVs), and to be used for application distribution in mass deployment scenarios typical of large enterprises.

The IBM Data Server Driver Package capabilities are as follows:

- ▶ Support for applications that use ODBC, CLI, or open source (PHP or Ruby) to access databases.
- ▶ Support for client applications and applets that are written in Java using JDBC, and for embedded SQL for Java (SQLJ).
- ▶ IBM Informix Dynamic Server support for .NET, PHP, and Ruby.
- ▶ Application header files to rebuild the open source drivers.
- ▶ Support for DB2 Interactive Call Level Interface (db2cli).
- ▶ On Windows operating systems, IBM Data Server Driver Package also provides support for applications that use .NET or OLE DB to access databases. In addition, this driver is available as an installable image, and a merge module is available to allow you to easily embed the driver in a Windows Installer-based installation.
- ▶ On Linux and UNIX operating systems, IBM Data Server Driver Package is not available as an installable image.



In Figure 2-9, an application uses the IBM Data Server Driver Package to access DB2 for z/OS data directly.

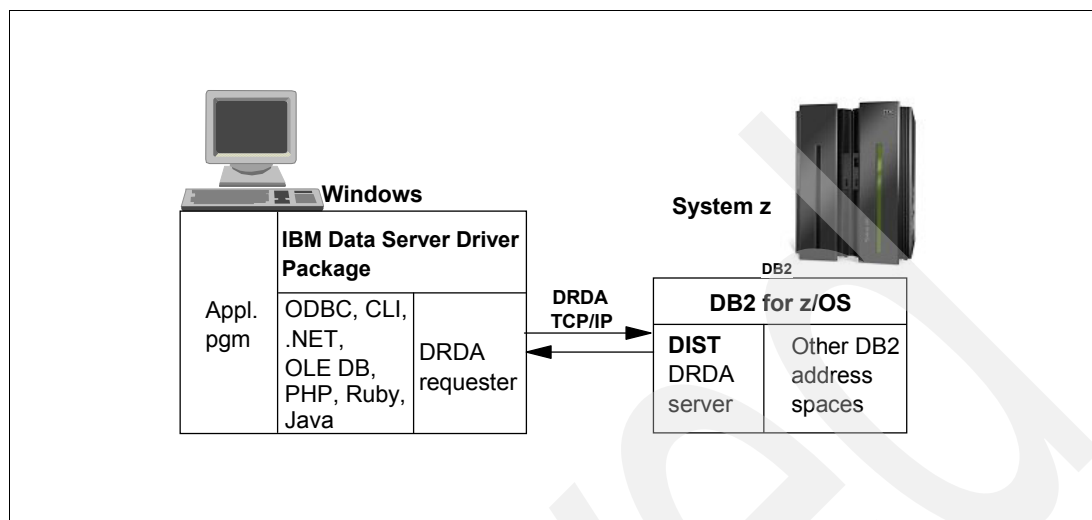


Figure 2-9 IBM Data Server Driver Package connecting directly to DB2 for z/OS.

## IBM Data Server Runtime Client

This product allows you to run applications on remote databases. Graphical user interface (GUI) tools are not included. Capabilities are as follows:

- ▶ Command line processor (CLP)
- ▶ Base client support for database connections, SQL statements, XQuery statements and commands
- ▶ Support for common database access interfaces (JDBC, SQLJ, ADO.NET, OLE DB, ODBC, command line interface (CLI), PHP and Ruby), including drivers and ability to define data sources
- ▶ Lightweight Directory Access Protocol (LDAP) exploitation
- ▶ Support for TCP/IP and Named Pipe
- ▶ Support for multiple concurrent copies and various licensing and packaging options

The IBM Data Server Runtime Client (Runtime Client) has a rich set of SQL APIs for deployment in more complex application environments. In Figure 2-10 the Runtime Client accesses DB2 for z/OS data directly, potentially supporting applications using different SQL APIs.

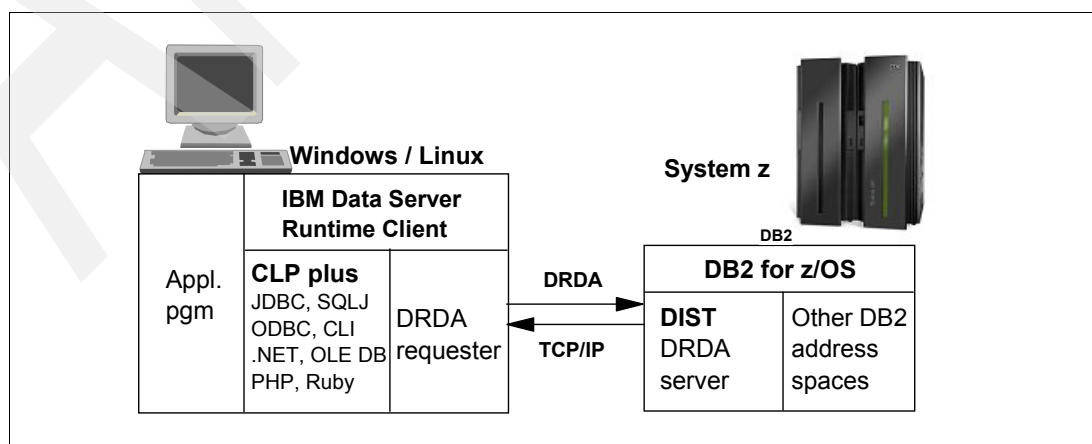


Figure 2-10 IBM Data Server Runtime Client connecting directly to DB2 for z/OS



## IBM Data Server Client

This is the full-function product for application development, database administration, and client/server configuration. Capabilities are as follows:

- ▶ Configuration Assistant
- ▶ Control Center and other graphical tools
- ▶ First Steps for new users
- ▶ Visual Studio® tools
- ▶ IBM Data Studio
- ▶ Application header files
- ▶ Precompilers for various programming languages
- ▶ Bind support
- ▶ All the functions included in the IBM Data Server Runtime Client

Figure 2-11 shows an example of the IBM Data Server Client supporting application development, database administration and applications with direct access to DB2 for z/OS data.

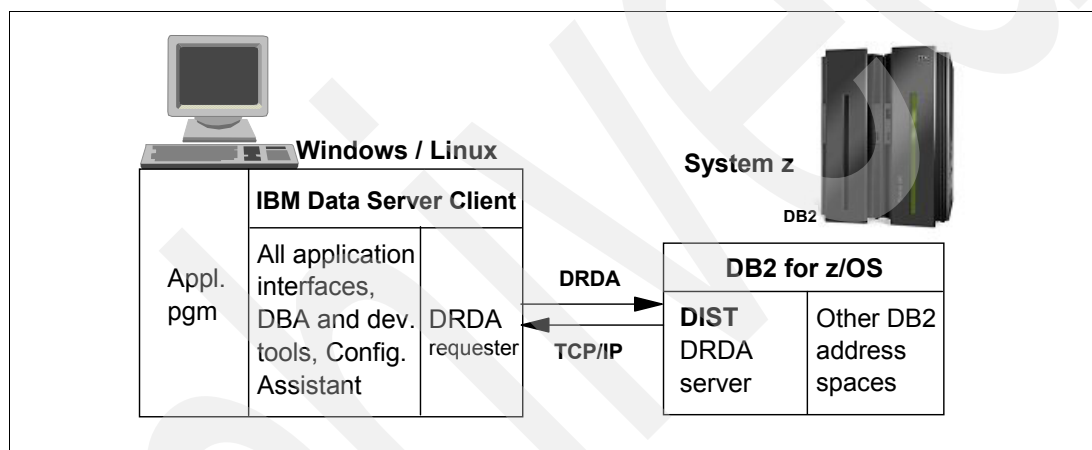


Figure 2-11 IBM Data Server Client connecting directly to DB2 for z/OS

## Driver and Client comparison

The highlights of the IBM Data Server products are summarized in Table 2-2. Refer to standard DB2 for LUW product documentation for additional details.

Table 2-2 IBM Data Server Drivers and Clients comparison

Product	Smallest footprint	JDBC and SQLJ	ODBC and CLI	OLE DB and .NET	Open Source	CLP	DBA, Dev, GUI tools
IBM Data Server Driver for JDBC and SQLJ	X	X					
IBM Data Server Driver for ODBC and CLI	X		X				
IBM Data Server Driver Package		X	X	X	X		
IBM Data Server Runtime Client		X	X	X	X	X	
IBM Data Server Client		X	X	X	X	X	X

### 2.3.3 Connecting to a DB2 data sharing group

Any of the IBM Data Server Drivers or Clients can connect your applications directly to a DB2 data sharing group. We show the basic configurations in Figure 2-12. Java-based applications use the Type 4 driver while non-Java-based applications use the CLI driver.

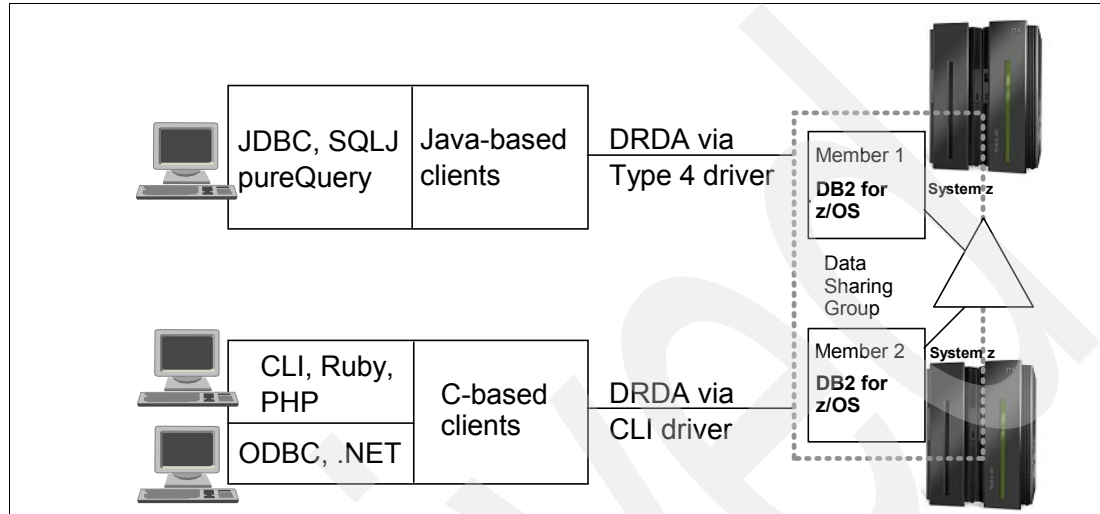


Figure 2-12 IBM Data Server Drivers connecting to a DB2 data sharing group

Java applications can balance their DB2 for z/OS data accesses across the data sharing group on transaction boundaries (that is, after commits). Java applications are thread-based. The Java driver can manage the threads across the driver's multiple connections to the data sharing members. The Type 4 driver providing connection concentration and workload balancing to distributed WebSphere applications accessing data in a DB2 for z/OS data sharing group provide the same support to individual Java applications.

IBM Data Server Drivers and Clients at Version 9.5 FixPack 4 behave the same way, with sysplex workload balancing (WLB) available on commit boundaries even on a single connection. Because FixPack 4 provides WLB for a single connection, DB2 Connect server is not needed for typical CLI/ODBC applications.

With IBM Data Server Drivers and Clients at Version 9.5 FixPack 3, individual ODBC and CLI applications tend to stay connected to the same member until they end, then drop the connection. Subsequent connections may be made to either member of the data sharing group. These applications exhibit this behavior because ODBC and CLI applications tend to be process-based and do not maintain a connection to the data sharing group once an individual process ends. While it is possible to write an ODBC process that manages separate connections to a data sharing group on behalf of multiple applications, this is not the standard implementation in most customer installations. Until you deploy FixPack 4, one way to take advantage of DB2 data sharing workload balancing benefits for single-connection ODBC and CLI applications is to use a DB2 Connect Server, as in Figure 2-20 on page 56.

ODBC and CLI applications that are part of a Web server configuration, for example in a .NET environment, can take advantage of transaction pooling and workload balancing across data sharing members, even though the individual applications will not tend to workload balance on their own behalf. The Web service performs the connection concentration and workload balancing function.

For more information about distributed access to a DB2 data sharing group, refer to Chapter 6, "Data sharing" on page 233.

## 2.3.4 Choosing the right configuration

For most customer configurations that currently use DB2 Connect Client for DRDA AR functions, one of the IBM Data Server products can replace the DB2 Connect Client, resulting in a significantly reduced footprint. A licence for DB2 Connect is still required.

There are many cases where customers currently use DB2 Connect Server to provide a single point of connectivity to numerous workstations supporting a variety of applications. In these cases, customers can deploy one or more of the IBM Data Server products with a smaller footprint and achieve overall benefits. For example, while DB2 Connect Servers provide a single point for managing connectivity, they introduce additional overhead and elapsed time to the applications accessing DB2 for z/OS data.

In addition, many customers must clone their DB2 Connect Servers to provide a fault tolerant configuration. Otherwise a server failure could impact a broad spectrum of business applications. The availability of an application on an individual workstation is not improved by the addition of a server. Even if the server is cloned, the availability of workstation access to DB2 for z/OS data does not improve over direct access.

Finally, if an application experiences a performance problem, the presence of the DB2 Connect Server complicates the efforts to identify the source of the problem. The overhead, elapsed time, cloned server, and performance monitoring challenges may be removed or reduced by implementing IBM Data Server products.

In the case of DB2 data sharing, the function available in the IBM Data Server Drivers and Clients is superior to the SYSPLEX support in the DB2 Connect Server. In a IBM Data Server Driver (or Client) configuration, if you enable sysplex workload balancing (sysplex WLB), which includes connection concentration, you will receive better availability characteristics than DB2 Connect Server with the SYSPLEX feature. The IBM Data Server Drivers and Clients provide seamless reroute after a network failure or a DB2 member fails or is stopped. In the DB2 Connect case, any time a member is shut down, the application gets a communications failure that it must handle.

In this section we illustrate several scenarios where IBM Data Server products can replace DB2 Connect Client or DB2 Connect Server. In the next section we describe a general situation in which a DB2 Connect Server may still be advantageous. Table 2-3 on page 48 provides a summary of the considerations for replacing DB2 Connect with the IBM Data Server Drivers or Clients.

Table 2-3 Replacing DB2 Connect Server with IBM Data Server Drivers or Clients

Pros	Cons
<p><b>Improved performance</b> Performance on the workstation or application servers or Web application servers can improve due to:</p> <ul style="list-style-type: none"> <li>► Reduced network traffic</li> <li>► Reduced code path</li> </ul> <p><b>Improved availability</b> Application access to DB2 for z/OS data equal to or superior to three-tier configuration due to elimination of a point of failure</p> <p><b>Improved visibility</b> Easier to monitor application, workstation, application server or Web application server traffic and behavior</p> <p><b>Improved problem determination</b> Easier to identify location of problems and customers affected. Tools for analyzing data and network traffic can provide detailed view</p> <p><b>Improved security</b> The z/OS Communications Server Intrusion Detection Services (IDS) can be used to control transports at the server.</p>	<p>Some reduction in control of workload priorities</p> <p>Limited ability to constrain low priority work</p> <p>Potential impact to high priority distributed or mainframe applications</p> <p>DB2 Connect Server required for XA transactions using multi-transport model</p>

### Business scenarios: A variety of requirements for distributed access

Most customers who use DB2 for z/OS to manage enterprise data have a variety of business applications with varying requirements for workstation clients. In some cases individual workstations deal directly with DB2 for z/OS data. In other cases business users access application servers which in turn access DB2 for z/OS data. Most customers will have Web application servers with Java-based applications. In addition there are the mainframe-based business applications, which also access DB2 for z/OS data, and which represent a substantial and vital element of business processing. These mainframe applications must also be considered in the overall effort to meet business requirements. This point will become important in a scenario in the next section.

### Current configuration with DB2 Connect

We show a typical customer configuration in Figure 2-13 on page 49, which might resemble your starting point when migrating to IBM Data Servers Drivers and Clients. Here the individual workstations access DB2 for z/OS directly using DB2 Connect Client installed on each workstation. (We use DB2 Connect Client to refer to the DB2 Connect product for use by an individual workstation.) The application servers and Web application servers use DB2 Connect Servers to take advantage of connection concentration and single point of management.

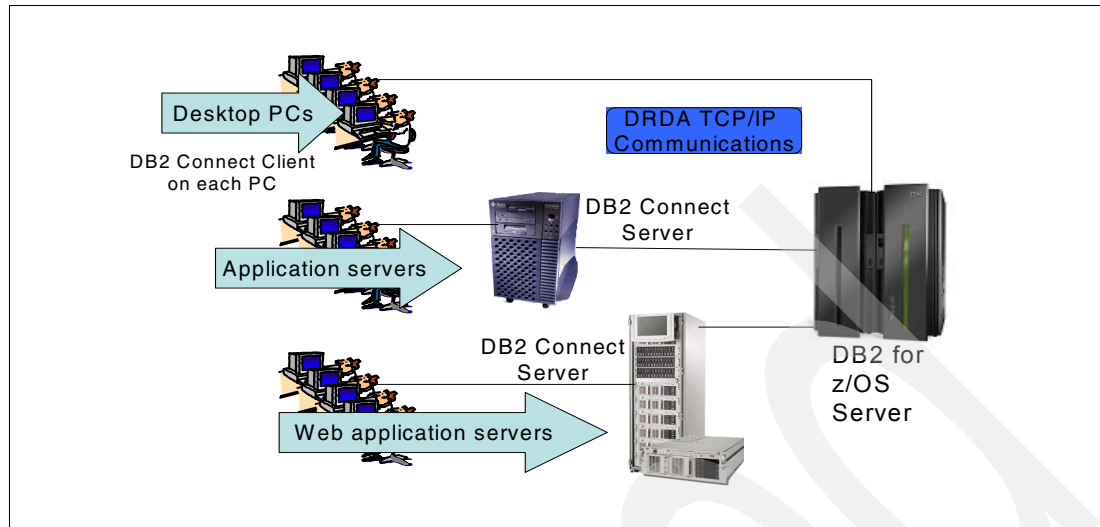


Figure 2-13 Current configuration with DB2 Connect Client and Server

### Replacing DB2 Connect Client and Server with IBM Data Server Drivers

For most workstations running business applications, especially those which require a single SQL API, the full range of DB2 Connect Client function is not necessary. The IBM Data Server Driver for ODBC and CLI and the IBM Data Server Driver for JDBC and SQLJ each provide the DRDA functions and support for the corresponding language APIs.

If you have developers and DBAs using workstations connected directly to DB2 for z/OS, you can install the IBM Data Server Client to provide the DRDA AR functions plus a rich set of productivity tools. If you have some workstations that require a CLP, you can install the IBM Data Server Runtime Client. For workstations that run business applications requiring more than a single API you can install the IBM Data Server Driver Package. Any of these options will require a smaller footprint than DB2 Connect Client.

You may also be able to replace the DB2 Connect Server with IBM Data Server Drivers. In cases where application servers only need to support ODBC and CLI APIs, the IBM Data Server Driver for ODBC and CLI will be sufficient. In cases where Web application servers only need to support JDBC and SQLJ APIs, the IBM Data Server Driver for JDBC and SQLJ will be sufficient. The IBM Data Server Driver Package may be necessary if the application servers or Web application servers need to support multiple APIs for the applications they serve.

If you require CLP support, or if your developers or DBAs are operating in these environments, you will need the IBM Data Server Runtime Client or the IBM Data Server Client. For example, if you need to compile your applications, you will need the IBM Data Server Client.

Figure 2-14 illustrates the situation where all the DB2 Connect Clients and Servers are replaced with one of the IBM Data Server Drivers or Clients.

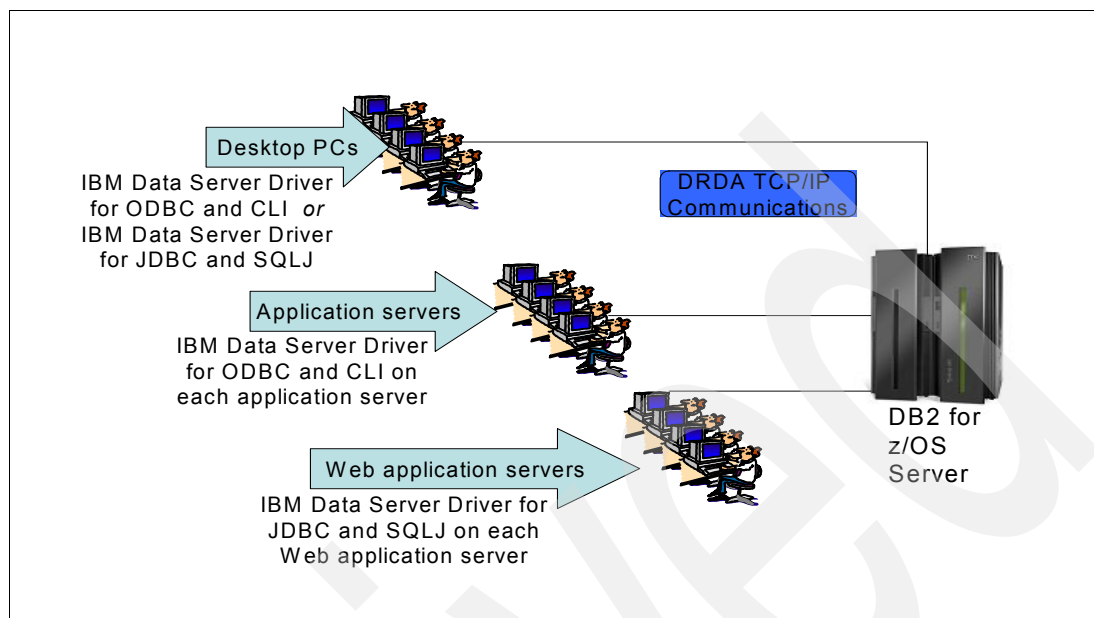


Figure 2-14 DB2 Connect Client and Server replaced with IBM Data Server Drivers

Each of the IBM Data Server products provides the DRDA AR functions necessary to access DB2 for z/OS data. In addition, they provide support for parallel sysplex, including workload balancing, to help you to achieve high application availability and workload balancing.

Refer to 2.7, “XA Support in DB2 for z/OS” on page 63 for discussion of a situation where DB2 Connect Server function is still required.

See Chapter 6, “Data sharing” on page 233 for more information about connecting to a DB2 data sharing group.

### 2.3.5 Ordering the IBM Data Server Drivers and Clients

You can download the IBM Data Server Drivers and Clients from the IBM download site (<http://www.ibm.com/support/docview.wss?rs=4020&uid=swg21385217>) where you will see Table 2-4 on page 51, which can help in identifying the package you need.

Table 2-4 IBM Data Server Client Packages: Latest downloads (V9.7)

Driver package	Description
IBM Data Server Driver Package (DS Driver)	This package contains drivers and libraries for various programming language environments. It provides support for Java (JDBC and SQLJ), C/C++ (ODBC and CLI), .NET drivers and database drivers for open source languages like PHP and Ruby. It also includes an interactive client tool called CLPPlus that is capable of executing SQL statements and scripts, and can generate custom reports.
IBM Data Server Driver for JDBC and SQLJ (JCC Driver)	Provides support for JDBC and SQLJ for client applications developed in Java. Supports JDBC 3 and JDBC 4 standard. Also called as JCC driver.
IBM Data Server Driver for ODBC and CLI (CLI Driver)	This is the smallest of all the client packages and provides support for Open Database Connectivity (ODBC) and Call Level Interface (CLI) libraries for the C/C++ client applications.
IBM Data Server Runtime Client	This package is a super-set of Data Server Driver package. It includes many DB2 specific utilities and libraries. It includes DB2 Command Line Processor (CLP) tool.
IBM Data Server Client	This is the all-in-one client package and includes all the client tools and libraries available. It includes DB2 Control Center, a graphical client tool that can be used to manage DB2 Servers. It also includes add-ins for Visual Studio.
IBM Database Add-Ins for Visual Studio	This package contains the add-ins for Visual Studio for .NET tooling support.

## 2.4 DB2 Connect to DB2 for z/OS: Past, present, and future

Many customers have large volumes of business-critical data stored in DB2 for z/OS databases. DB2 for z/OS is an excellent database for business-critical data because of the outstanding capacity and availability characteristics of the mainframe platform. Historically, DB2 Connect was the product for Windows, UNIX, and Linux platforms that provided the DRDA requester functions to allow the strengths of DB2 for z/OS to be combined with the applications and development frameworks available on distributed platforms. DB2 Connect provided these DRDA requester functions in two-tier, three-tier (or server), or application server configurations. Using DB2 Connect, any application on any platform that could interface with DB2 Connect could communicate with any DRDA AS or DS.

In this section we describe configurations with DB2 Connect as a requester and DB2 for z/OS as a server. We give examples of DB2 Connect in two-tier, three-tier, and application server configurations. These examples may be similar to what you have used in the past or what you are using now to access DB2 for z/OS data. We discuss DB2 Connect Client and DB2 Connect Server functions only. There are a variety of licensing alternatives. Refer to standard DB2 Connect product documentation for details, starting at the DB2 Connect Web site:

<http://www.ibm.com/software/data/db2/db2connect/>

The IBM strategy is to remove the reliance on the DB2 Connect modules and replace DB2 Connect with the IBM Data Server Drivers or Clients. While DB2 Connect licenses (in the form of DB2 Connect license files) are still required, you can replace DB2 Connect modules with the IBM Data Server Drivers or Clients and receive equivalent or superior function. In addition, you can reduce complexity, improve performance, and deploy application solutions with smaller footprints for your business users.

We intend for the descriptions in this section to correspond to the recent past or what you have currently installed. In the previous section, which represents the future, we provide descriptions and examples of the IBM Data Server Clients and Drivers. Refer to 2.3, “IBM Data Server Drivers and Clients as requesters” on page 40 if you are not familiar with the IBM Data Server Drivers and Clients.

## 2.4.1 DB2 Connect Client as requester

DB2 Connect Client provides direct access from workstations to mainframe DB2 servers. This product is only licensed to a single person, so it cannot be used for server configurations. Each workstation has to install the DB2 Connect Client and has to be configured with network connectivity. DB2 Connect Client is used for database access by applications implementing a two-tier architecture. DB2 Connect Client at Version 9.5 FixPack 3 is available for Linux, UNIX, and Windows platforms.

In Figure 2-15 we show that application programs running on a workstation with DB2 Connect Client can use one of numerous APIs to access data stored in DB2 for z/OS.

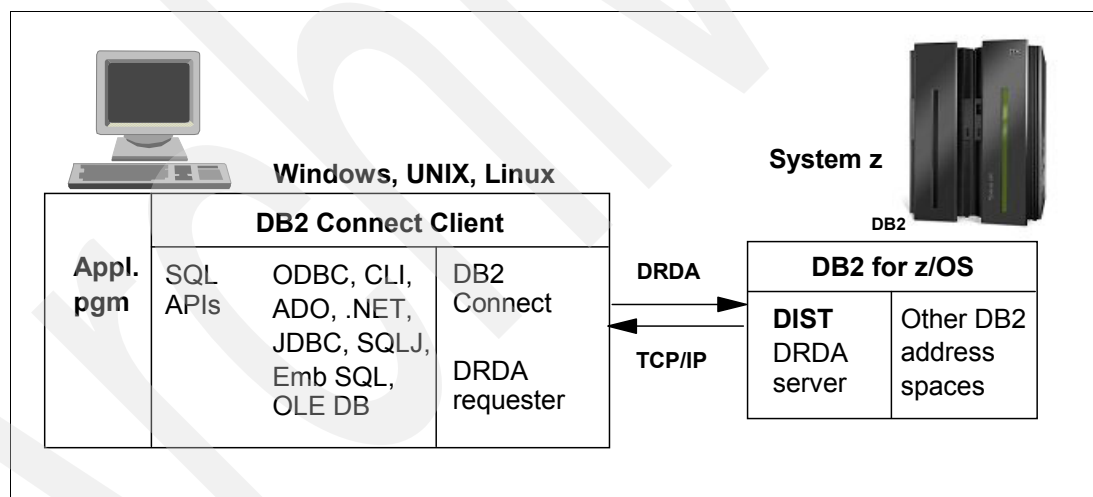


Figure 2-15 DB2 Connect Client connecting to DB2 for z/OS

This example suggests the full functionality of DB2 Connect Client, all of which is available to each workstation that installs it. For many business situations you may prefer a smaller code footprint that still enables workstation applications to access DB2 for z/OS data. The IBM Data Server Drivers offer the DRDA AR functions without requiring the installation of DB2 Connect. A DB2 Connect license is still required. Refer to 2.3, “IBM Data Server Drivers and Clients as requesters” on page 40 for more information.



## 2.4.2 DB2 Connect Server

DB2 Connect Server is a server version of DB2 Connect Client. DB2 Connect Server provides all the functions of DB2 Connect Client, plus connection concentration, enhanced database support, sysplex workload balancing, and performance, monitoring, and availability features. DB2 Connect Server supports three-tier and application server configurations.

In a three-tier configuration, where multiple workstations access DB2 for z/OS data using a DB2 Connect Server, each workstation must have a driver or client installed. You can select from the following drivers or clients:

- ▶ IBM Data Server Client
- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Driver Package
- ▶ IBM Data Server Driver for ODBC and CLI
- ▶ IBM Data Server Driver for JDBC and SQLJ

In a three-tier configuration, DB2 Connect Server generally provides the DRDA AS function on behalf of one or more of the drivers or clients listed above, and DB2 for z/OS provides the DRDA Data Server (DS) function.

The IBM Data Server Driver or Client (DRDA AR) provides the SQL interfaces and passes the database request through the DB2 Connect Server (DRDA AS) to the DB2 for z/OS (DRDA DS). Figure 2-16 shows a DB2 Connect Server providing access to DB2 for z/OS data for a number of desktop clients. These desktop clients could have a variety of IBM Data Server Drivers or Clients installed. Refer to 2.3, “IBM Data Server Drivers and Clients as requesters” on page 40 for a discussion of these choices.

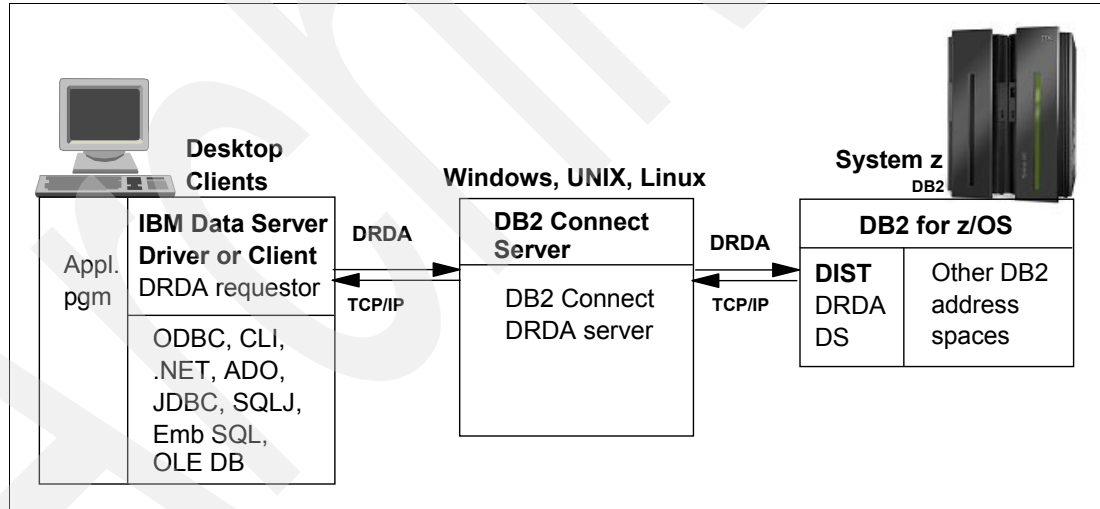


Figure 2-16 DB2 Connect Server providing access to DB2 for z/OS

Beginning with the Type 4 driver for Java and for ODBC and CLI with the IBM Data Server Drivers and Clients Version 9.5 FixPack 3, DB2 Connect Server is no longer required (although a DB2 Connect license file is required). One of the historic advantages of the DB2 Connect Server configuration over the DB2 Connect Client configuration was that the server could provide the connection concentrator function. The server could manage a relatively small number of connections (or transports) to DB2 for z/OS on behalf of a large number of workstation applications. Figure 2-17 on page 54 illustrates an expansion in the number of desktop clients without a corresponding increase in the number of connections between the DB2 Connect Server and DB2 for z/OS. Refer to 1.7.3, “Transaction pooling” on page 25 for the introduction to connection concentration.

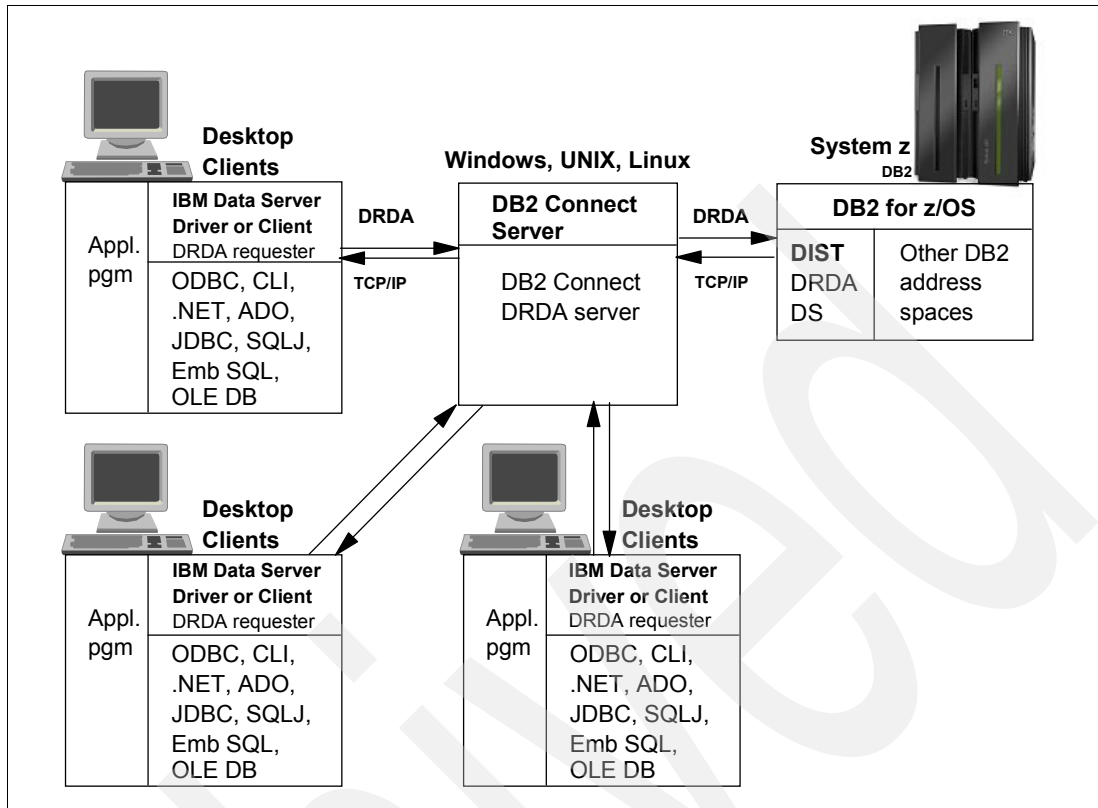


Figure 2-17 Example of DB2 Connect Server providing connection concentration

There may be some situations where a DB2 Connect Server provides you with advantages in a configuration similar to Figure 2-17. This will be the exception, not the rule. For a discussion of such a situation, refer to 2.5, “DB2 Connect Server on Linux on IBM System z” on page 58. As a rule, you should connect your IBM Data Server Drivers or Clients directly to DB2 for z/OS.

Many applications are built using a three-tier model. They can be purchased applications like SAP®, Siebel®, or PeopleSoft®, or home-grown applications using Java-based or .NET development frameworks. An example is shown in Figure 2-18.

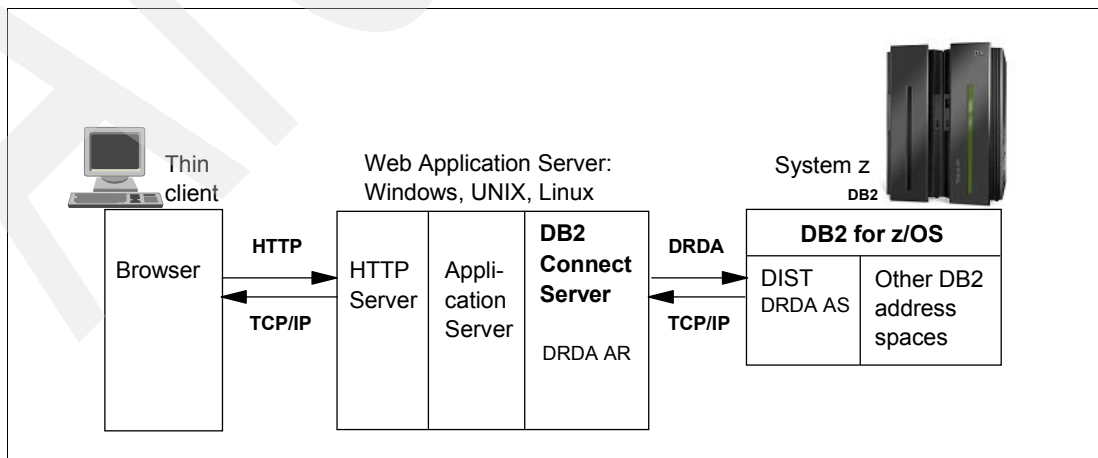


Figure 2-18 DB2 Connect Server in a Web application server environment

In this model, the application acts as a “concentrator” to a thin client (typically a browser), which means there is no need to deploy DB2 clients. DB2 Connect Server has provided the DB2 connectivity from the application server to the database server on DB2 for z/OS. Now you can use the IBM Data Server Drivers or Clients to connect the Web application server to DB2 for z/OS.

### 2.4.3 Connecting to a DB2 data sharing group from DB2 Connect

DB2 data sharing provides capacity, scalability, availability, and workload balancing benefits to applications that use DB2 for z/OS data. If you use DB2 Connect to access data in a DB2 data sharing group, DB2 Connect should specify the group dynamic virtual IP address (group DVIPA), the SQL port and the location name of the DB2 for z/OS data sharing group. The Sysplex Distributor will direct the request to an available DB2 member. After the first connection, DB2 Connect will receive a server list from DB2 for z/OS that indicates which members of the data sharing group are available. Subsequent requests can be balanced on transaction boundaries among the members of the data sharing group.

Refer to Chapter 6, “Data sharing” on page 233 for more information about Sysplex Distributor, DVIPA and configuring DB2 data sharing for availability and workload balancing.

#### DB2 Connect Client as requester to a data sharing group

In Figure 2-19 we show DB2 Connect Client connecting to the members of a DB2 data sharing group. Applications connect to the data sharing group (by the group’s LOCATION) and DB2 Connect manages transports to the members of the group and can balance work between the members across transaction boundaries (after commits).

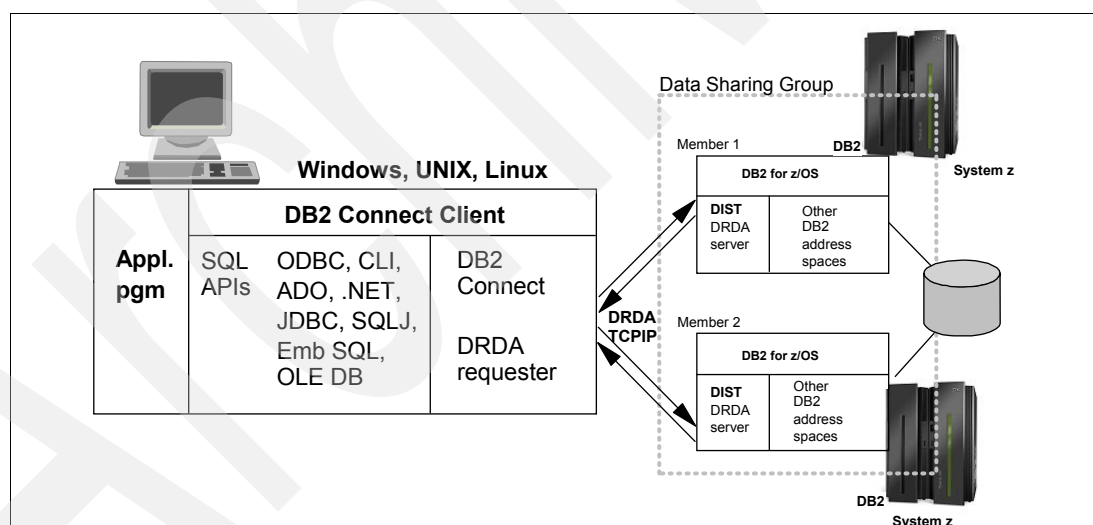


Figure 2-19 DB2 Connect Client connecting to a DB2 data sharing group

#### DB2 Connect Server connecting to a data sharing group

In Figure 2-20 on page 56 we show a DB2 Connect Server allowing multiple desktops to access the members of a DB2 data sharing group. After the first connection from DB2 Connect Server, subsequent connections are balanced across the available members of the data sharing group. Application requests to DB2 for z/OS can be balanced across the DB2 data sharing members on transaction boundaries (after commits).

DB2 Connect Server provides connection concentration for the desktop clients, optimizing DB2 resource usage in the parallel sysplex.

You may have this configuration currently. Often the DB2 Connect Server represents a single point of failure to the desktop clients. By using the IBM Data Server Drivers to connect directly to the DB2 for z/OS data sharing group, you may be able to improve the overall availability of your applications.

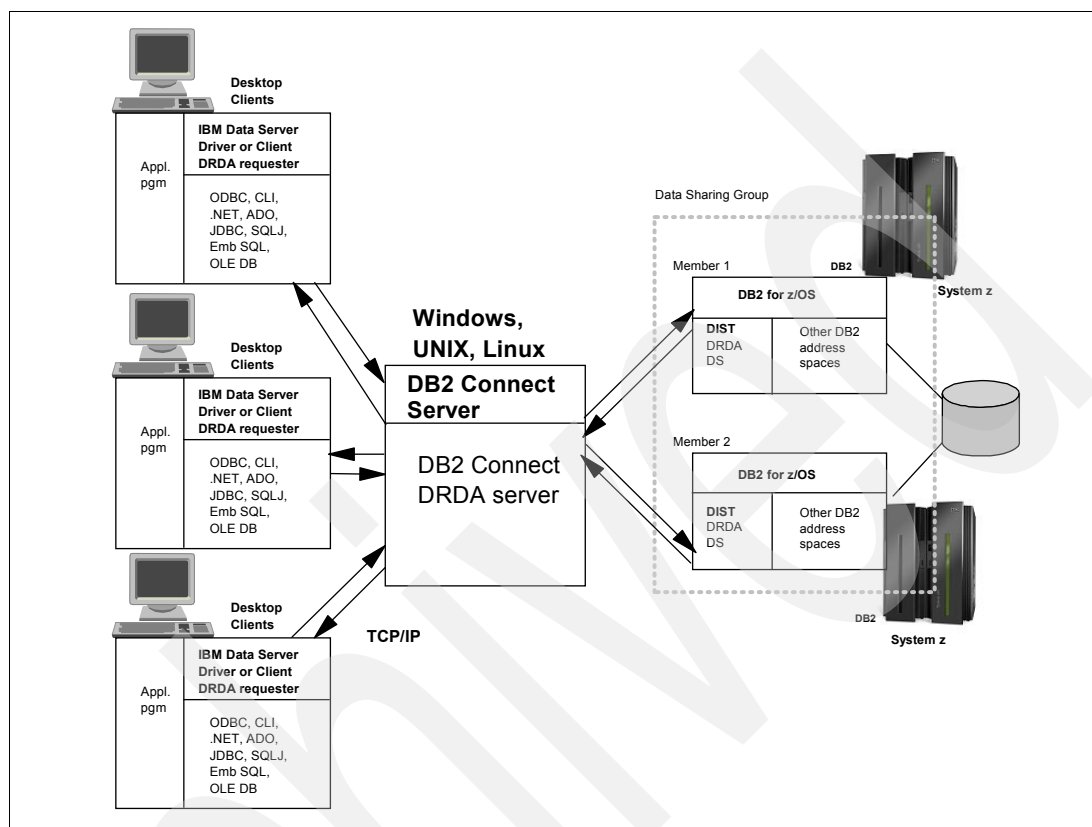


Figure 2-20 DB2 Connect Server providing access to a DB2 data sharing group

## 2.4.4 DB2 Connect: A case of managing access to DB2 threads

There are rare situations where you may need to implement controls on distributed access to DB2 for z/OS. One of the controls you might choose is a DB2 Connect Server to manage the number of connections for the workstation, application server, or Web application server traffic to DB2 for z/OS even when the IBM Data Server products provide all the connectivity functions you require. These rare situations would be those where there are a number of high priority clients and a smaller number of lower priority clients who might require extended connectivity to DB2. For example, the high priority business tasks could be efficient transactions that commit after each DB2 interaction, freeing the DB2 DBAT resources, while the lower priority business tasks might require several interactions, and network round-trips, before the task is complete. In such a situation, a sporadically large number of lower priority tasks might accumulate enough DBATs to constrain some higher priority business tasks from acquiring a DBAT.

Distributed access requests to DB2 for z/OS are managed, once the thread is established within DB2, by z/OS Workload Manager (WLM) service classes. Neither DDF nor WLM make a distinction between the various sources of distributed access requests for connection to DB2 nor for allocation of active database threads (DBATs). Thus, requests that are low in priority from a business perspective compete for DB2 connections and DB2 DBATs equally with high priority business requests. Once a DBAT is running, it will receive access to DB2 resources, but its access to CPU service units will be based on WLM service classifications.

A low priority business transaction could receive a smaller share of service units and therefore retain its DBAT and other DB2 resources for a longer time than a high priority business transaction that receives a larger share of service units, completes its processing quickly, and then releases the DB2 resources it used.

One possible effect is that a large number of low priority requests, especially complex ones, could reduce the number of or monopolize the available DBATs, effectively delaying high priority work from entering the system. If the low priority requests are of long duration, they could shut out some high priority work requests for an extended period. Since there is no way to specify within DDF which requests are high priority and which are low priority, you must implement your distributed architecture with due consideration to the relative priorities of the various requests.

The effect described above can apply to local agents attached to DB2 for z/OS as well. Low priority requests for distributed access may accumulate and retain DB2 for z/OS resources, such as locks, EDM Pool storage, or sort work areas, in such a way as to impact CICS, IMS or WebSphere for z/OS transactions.

IBM's direction is to provide centralized client management or server controls to mitigate the effect we describe above. The current alternatives are to implement network based controls, to use Trusted Context, or to use DB2 Connect Server. Network based controls include z/OS Communications Server functions such as Intrusion Detection Services (IDS), which can use traffic regulation to control the number of connections allowed from a requester. These require that network administrators understand the DB2 for z/OS and distributed environment to mitigate DB2 constraints in threads. One limitation of this alternative is that the controls are based on IP address, and you may have some clients who do not have a dedicated IP address which you can use to establish the desired control. Trusted Context could be implemented to address what requests have access, but may not be the best choice for managing priority. The alternative that may apply in some situations is to use DB2 Connect Server to manage access to DB2 threads. See Figure 2-21.

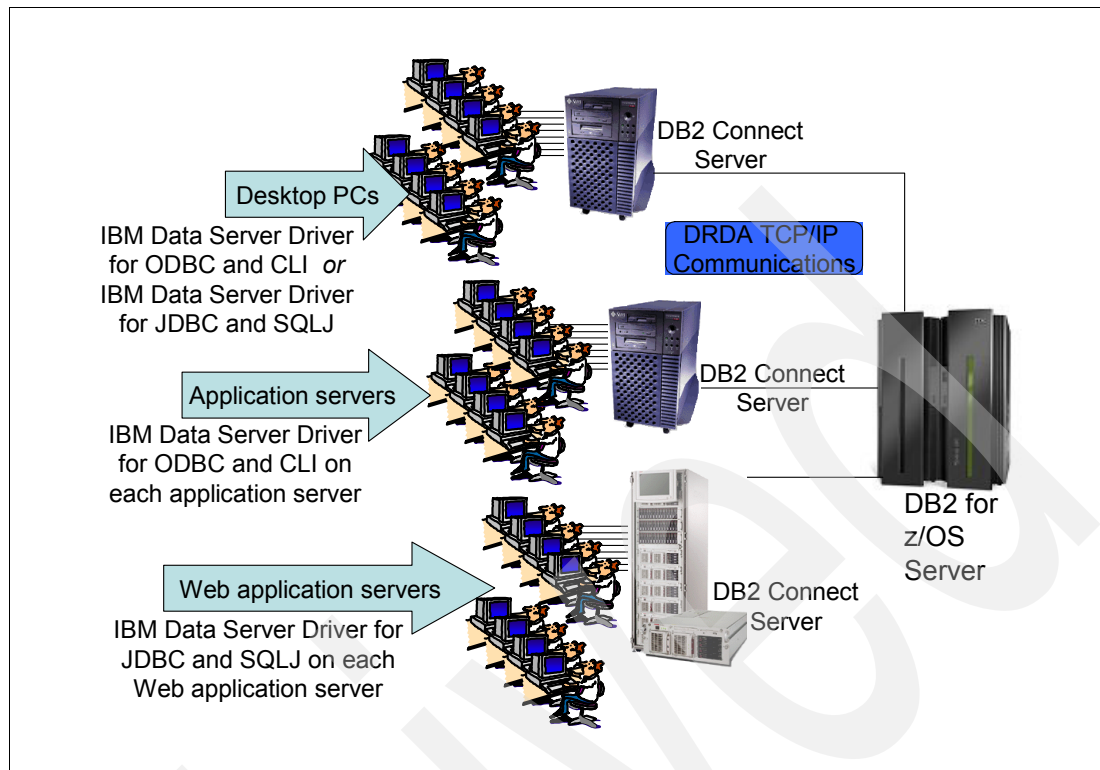


Figure 2-21 Controlling DB2 threads with a DB2 Connect Server

You can use DB2 Connect Server to control application access to DB2 for z/OS connection and DBAT resources, and you can reduce the likelihood that low priority requests will block access for high priority requests. You can accomplish this by assigning lower priority client requests to a DB2 Connect Server that has few transports, or connections, defined to DB2 for z/OS. Thus, applications representing high priority requests could have access to more connections and DBATs, and low priority requests could be grouped together with fewer connections and DBATs.

## 2.5 DB2 Connect Server on Linux on IBM System z

You may have a configuration where a DB2 Connect Server is installed on Linux on IBM System z (Linux on z) and connects to DB2 for z/OS on another LPAR on the same System z machine using HiperSockets™. There are some advantages to this configuration, based on the strengths of the System z platform, the security of the connection between DB2 Connect and DB2 for z/OS, and the performance characteristics of HiperSockets. The System z platform strengths include environmental, operational, and availability characteristics. The security and performance benefits are based on the use of HiperSockets instead of external network connections.

### 2.5.1 DB2 Connect on Linux on z with HiperSockets

The use of HiperSockets is illustrated in Figure 2-22. In the diagram, the two bottom layers of the OSI model are replaced by the HiperSocket. It shows two system images in one System z machine: DB2 Connect Server on Linux on z, and a DB2 for z/OS subsystem. DB2 Connect and DB2 for z/OS can communicate using TCP/IP over the HiperSockets. As is clear in this figure, there are no external network cables between DB2 Connect and DB2 for z/OS. The

network traffic is through microcode at internal memory copy speed. This can significantly reduce application elapsed time for clients connecting to DB2 for z/OS through a DB2 Connect Server. For more information about HiperSockets, see *HiperSockets Implementation Guide*, SG24-6816.

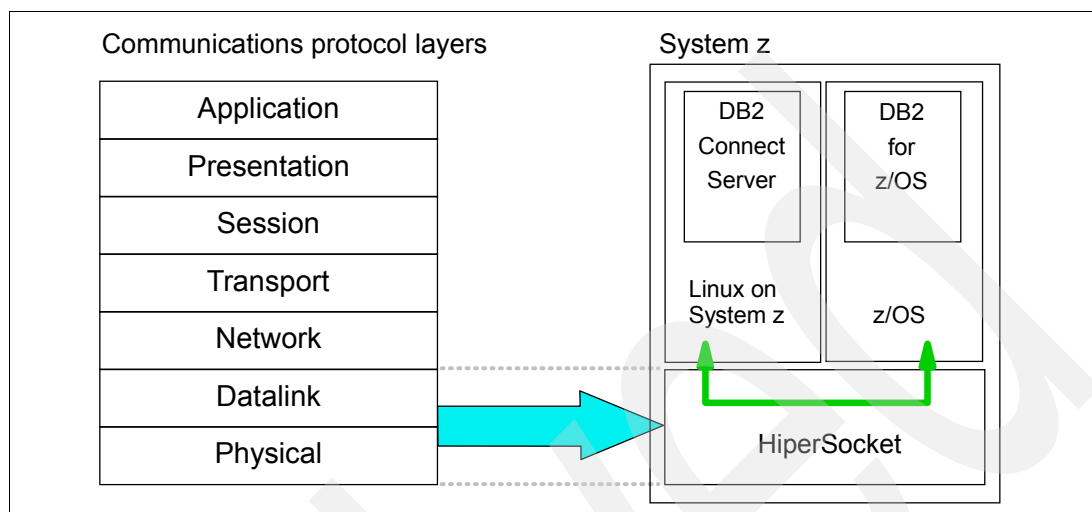


Figure 2-22 HiperSocket: Example of multiple LPAR communication

In cases where you choose to implement DB2 Connect Server, you can locate the DB2 Connect Server on the same System z machine with DB2 for z/OS and achieve the following advantages over DB2 Connect Server installed on a separate machine:

- ▶ **Elimination of middle-tier hardware**  
Many customers using DB2 Connect Servers have multiple separate systems or machines to house the servers. By consolidating the servers in Linux on System z, these customers can reduce the cost of server support, reducing the total cost of ownership (TCO). Refer to Figure 2-23 on page 60 for an illustration of this scenario.
- ▶ **Fast data transfer between DB2 Connect and DB2 for z/OS through HiperSockets**  
The network cable between both systems has been removed, reducing network latency and improving application response time.
- ▶ **Increased security**  
There are no wires between DB2 Connect and DB2 for z/OS, so one less network connection is susceptible to physical access.



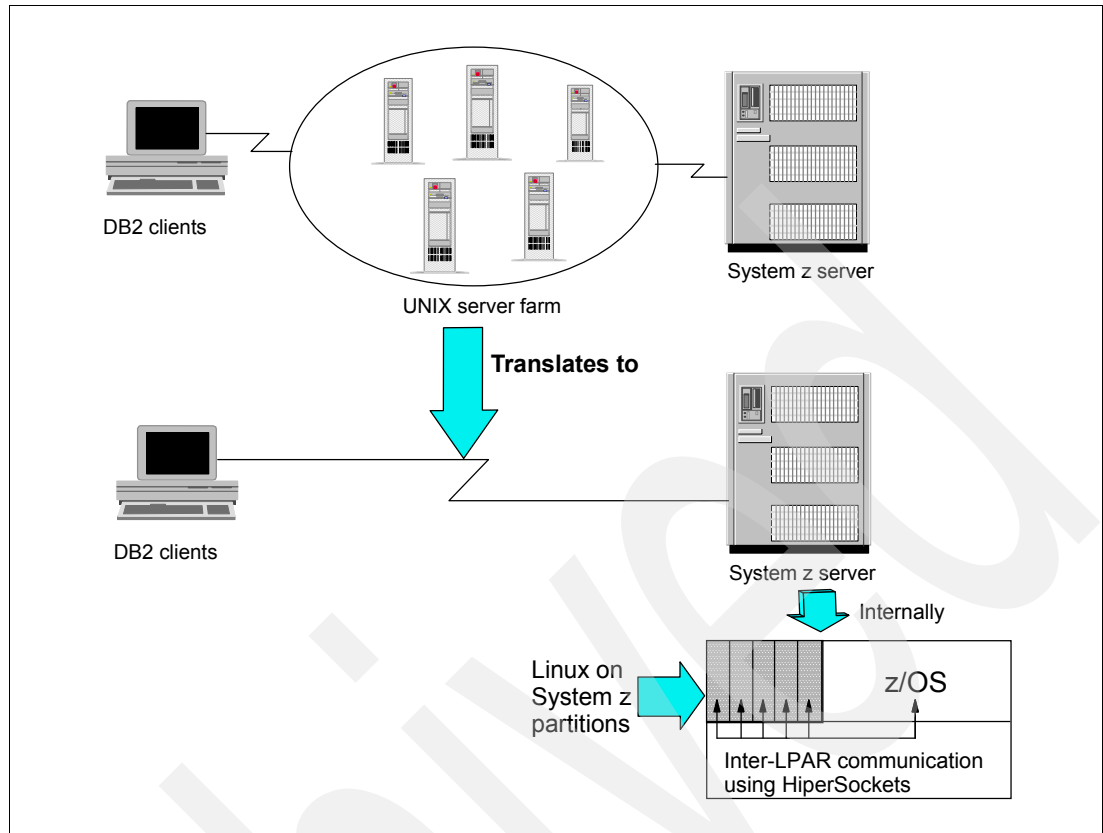


Figure 2-23 Server consolidation and HiperSockets and Linux on System z for DB2 Connect

In addition, you can create additional system images without bringing any new system devices into your machine room. This configuration provides flexibility to your system. We show an example of HiperSockets with DB2 Connect Server and DB2 for z/OS supporting multiple client types in Figure 2-24 on page 61.

In this example, DB2 for z/OS serves as a robust database server and DB2 Connect acts as an open server to Web application servers and fat clients. HiperSockets are used as a fast communication link providing TCP/IP socket communication.



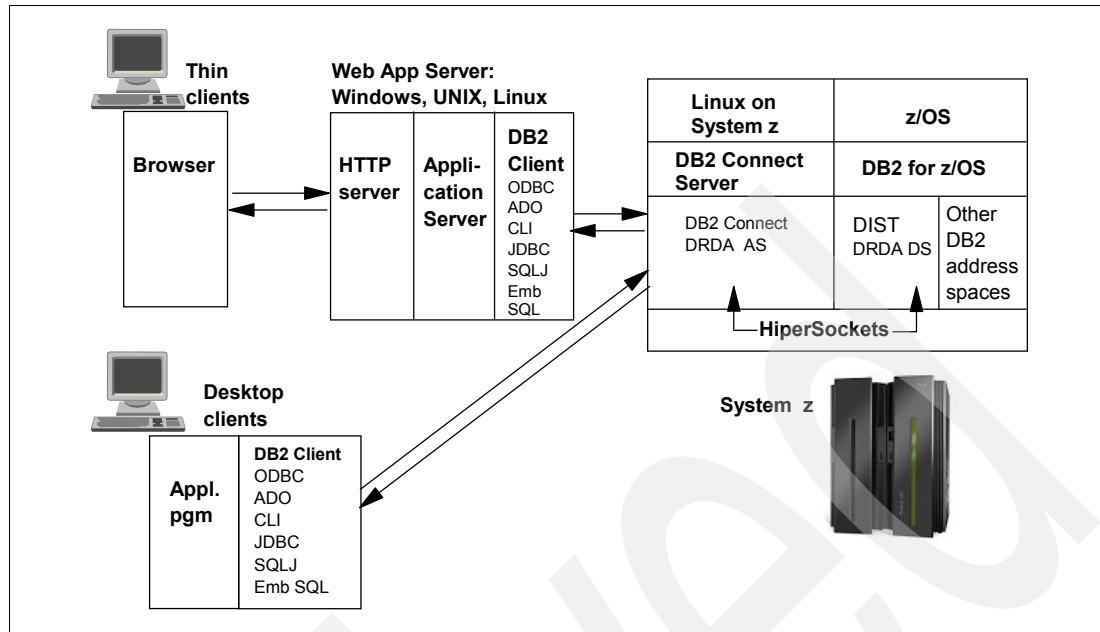


Figure 2-24 HiperSocket: DB2 Connect using HiperSocket to communicate with DB2 for z/OS

## 2.5.2 HiperSockets and DB2 data sharing configurations

The benefits to using HiperSockets between DB2 Connect Server on Linux on z and DB2 for z/OS are clear in the case of a single DB2 for z/OS subsystem. Some of those benefits accrue in a DB2 data sharing environment as well. But there are special considerations with using HiperSockets in a DB2 data sharing environment. There are different HiperSocket implications depending on whether you are using DB2 for z/OS V8 or DB2 9 for z/OS. This section discusses these implications. In either case, workload balancing algorithms do not take network speed into consideration, which may reduce the overall benefit of HiperSockets in a data sharing environment.

Among the benefits most customers seek when they implement DB2 data sharing are the following:

- ▶ Flexible capacity growth and scalability
- ▶ High availability of data for business applications
- ▶ Dynamic workload balancing

Requirements for high availability distributed access to a DB2 data sharing group include implementing Sysplex Distributor, dynamic virtual IP addressing (DVIPA) for the members of the data sharing group, and group DVIPA (or distributed DVIPA) for the data sharing group. Refer to 3.1.5, “Sample DB2 data sharing DVIPA and Sysplex Distributor setup” on page 79 for information about best practices for distributed access to data sharing groups.

In DB2 for z/OS V8, meeting these requirements means binding the DB2 members to specific IP addresses in the TCP/IP definitions. Binding members to specific addresses allows requests to be routed to a DB2 member even if the member is restarted on an LPAR other than the LPAR on which that member normally runs. On the other hand, binding the DB2 members to specific addresses makes it more difficult to define the connections from the DB2 Connect Server on Linux on z. Because the data sharing member on the same system is listening to its bound address, DB2 Connect must specify that IP address. A TCP/IP route table definition is needed to direct the DB2 Connect traffic for the data sharing member's IP address over the HiperSocket. In the event the DB2 data sharing member is restarted on

another LPAR, the IP route table should be dynamically updated with that information. This maintains the high availability benefits of data sharing for applications using DB2 Connect Servers on Linux on z.

In DB2 9 for z/OS, it is not required that DB2 bind to specific addresses. Instead you can define the IP addresses in the DB2 bootstrap data set (BSDS). In this case, DB2 can accept connection requests on any IP address. This will ease the definition of connections from DB2 Connect Server on Linux for System z to DB2 for z/OS. DB2 Connect will not have to specify both the IP address to reach the DB2 for z/OS member across the HiperSocket.

The above discussion addresses the high availability benefits of data sharing in a DB2 Connect Server on Linux on z environment. We now turn to workload balancing and capacity benefits. Most customers who implement HiperSockets between DB2 Connect Server and a member of a data sharing group would like most of the traffic to use the HiperSocket, to gain the benefits we described above in 2.5.1, “DB2 Connect on Linux on z with HiperSockets” on page 58. However, the workload balancing algorithms in the Sysplex Distributor and DB2 will not take the speed of the HiperSocket into consideration. Refer to Figure 2-25 for an illustration of this situation.

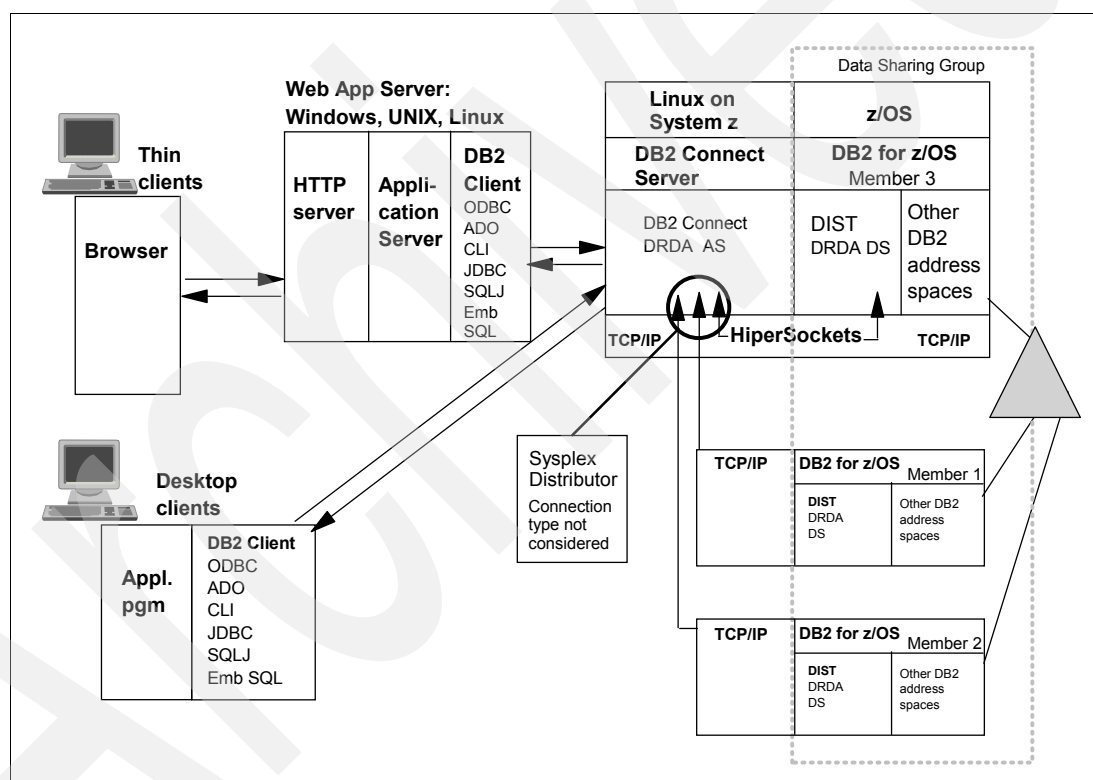


Figure 2-25 HiperSockets in a data sharing environment

On the right side of this diagram are three DB2 members in a data sharing group. One of them resides on the same System z machine with DB2 Connect Server running in a Linux on z environment. When DB2 Connect Server requests a connection to a member of the DB2 data sharing group, and the data sharing best practices have been followed, the connection request goes to the Sysplex Distributor application. Sysplex distributor will determine which member of the data sharing group is available, then route the request to that member. Sysplex distributor will spread the connection requests across the members without considering the special characteristics of the HiperSocket. Initial connection requests will be balanced approximately evenly, depending on the relative state of the various DB2 members. This means that in a two-member DB2 data sharing group, with one member on the same

System z machine with DB2 Connect Server on Linux on z, only half of the connection requests will end up taking advantage of the HiperSocket. In a three-way data sharing group, where only one member is on the same System z machine with DB2 Connect Server, about one third of the initial requests would end up using the HiperSocket.

After the initial connections are made, DB2 Connect Server can workload balance across the DB2 members based on the server list returned after each new DB2 for z/OS transaction. The workload balancing decision does not consider the existence of the HiperSocket. This results in the same load distribution described above; approximately half of the traffic will use the HiperSocket in a two-member data sharing group, approximately one third in a three-member group, and so on. This behavior is the same in DB2 for z/OS V8 and in DB2 9 for z/OS. This means customers may not achieve the full application elapsed time benefit they anticipated from HiperSockets.

One possible response to this is to eliminate the DB2 Connect Server from the configuration by implementing the IBM Data Server products as described in 2.3, “IBM Data Server Drivers and Clients as requesters” on page 40. This removes the code path of the DB2 Connect Server while maintaining the data sharing advantages of high availability, workload balancing, and flexible capacity and scalability.

Another possible response to this is to continue to use HiperSockets because of the server consolidation benefits, to realize the data sharing benefits, and to realize only part of the network latency benefit.

## 2.6 DB2 for z/OS requester: Any (DB2) DRDA server

DB2 for z/OS can be a DRDA AR to any DRDA-enabled data source. Within the IBM Information Management family, the following DBMSs are accessible:

- ▶ z/OS  
DB2 for z/OS V8 or DB2 9 for z/OS
- ▶ AIX, Linux, UNIX and Windows  
DB2 for LUW 9.7, 9.5 or 9.1
- ▶ VSE and VM  
DB2 Server for VSE and VM 7.3 or higher
- ▶ IBM i (or i5/OS)  
DB2 for i 6.1 (or DB2 for i5/OS 5.4)
- ▶ AIX, UNIX, Solaris  
Informix Dynamic Server 11.50 or 11.10

## 2.7 XA Support in DB2 for z/OS

In DB2 for z/OS Version 8, DB2 supported XA flows between the JCC Type 4 driver and DB2 for z/OS. A WebSphere Application Server could use the Type 4 driver to include DB2 for z/OS data in an XA transaction. A non-Java-based application that wanted to include DB2 for z/OS data in an XA transaction had to use DB2 Connect. DB2 Connect provided a translation between the XA flows and DRDA flows. Refer to Figure 2-26 on page 64 for an illustration of this configuration.

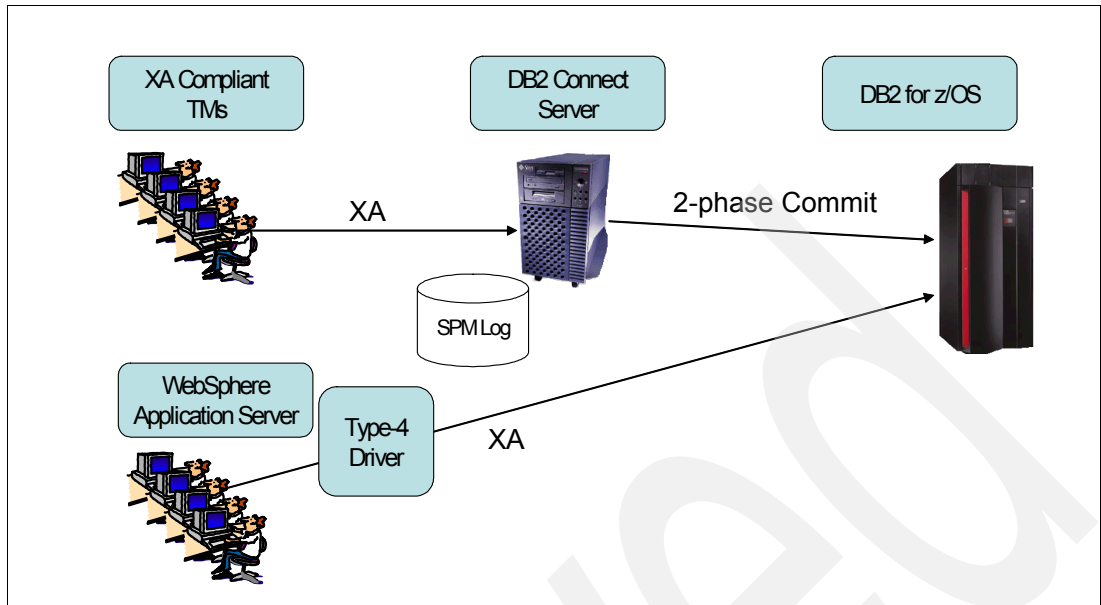


Figure 2-26 XA transaction support with DB2 Connect or WebSphere Application Server.

With the recent support provided by the IBM Data Server Driver and Client products at Version 9.5 FixPack 3, and with the corresponding PTFs for APAR PK69659 installed on DB2 for z/OS, clients using either the Java-based drivers or the non-Java-based drivers can include DB2 for z/OS data in XA transactions while connecting directly to DB2 for z/OS. We illustrate this configuration in Figure 2-27.

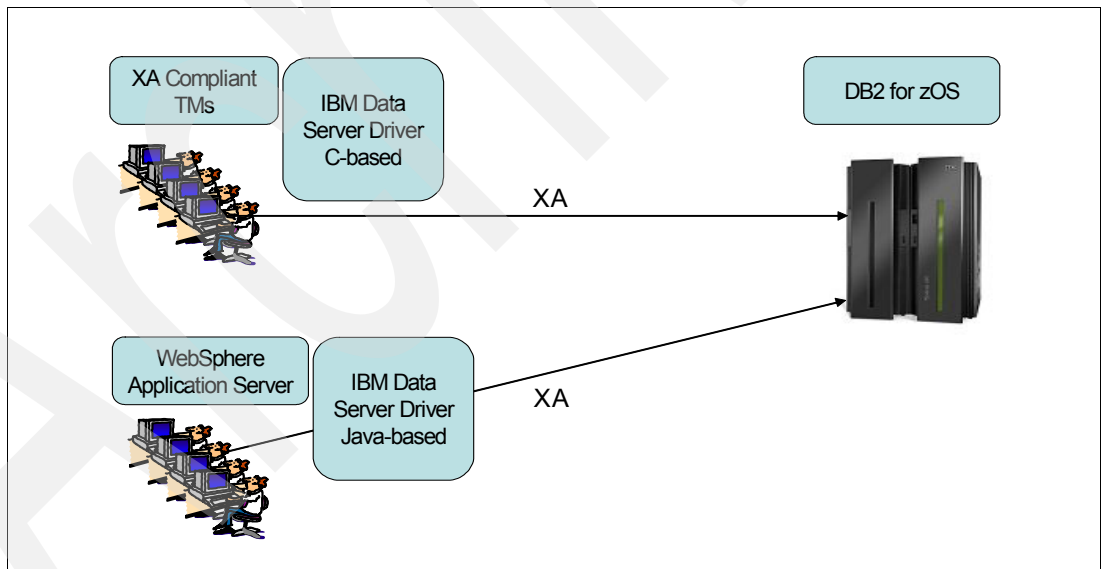


Figure 2-27 XA transaction support without DB2 Connect

The applications shown in the upper part of the diagram can use any of the drivers that support ODBC and CLI, depending on their operating environment:

- ▶ IBM Data Server Driver for ODBC and CLI
- ▶ IBM Data Server Driver for ODBC, CLI and .NET (Windows), currently replaced by the IBM Data Server Driver Package.
- ▶ IBM Data Server Driver for ODBC, CLI and Open Source (AIX, Linux or UNIX), currently replaced by the IBM Data Server Driver Package.
- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Client

The lower part of the diagram shows Java-based applications in a WebSphere Application Server environment. These applications could be written with JDBC or SQLJ and could be any Java-based application, not only those running in WebSphere Application Server. These applications could use the IBM Data Server Driver for JDBC and SQLJ, the IBM Data Server Runtime Client, or the IBM Data Server Client.

Your DB2 for z/OS may be in a data sharing environment. XA transactions are still supported, but in the case of indoubt resolution, the XA transaction may be routed to a member other than the one with which the transaction was originally communicating. The member then retrieves the XID from the Shared Communications Area (SCA) in the coupling facility to route the resolution to the correct member. We show this configuration in Figure 2-28.

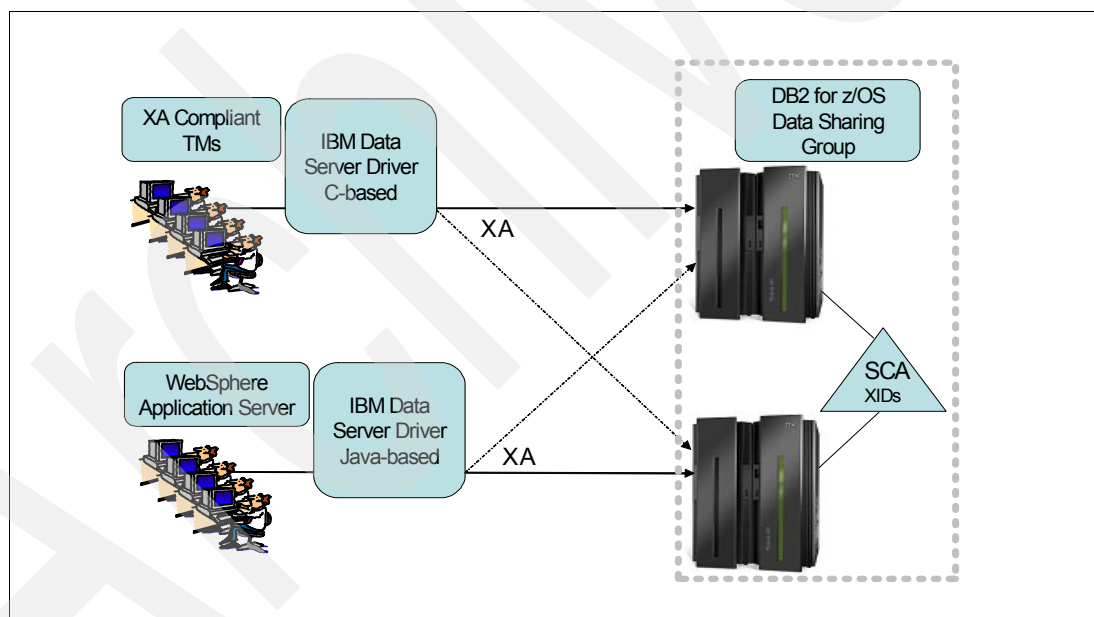


Figure 2-28 XA transaction support in a DB2 data sharing environment

The XA multi-transport model is not supported with direct connect to DB2 for z/OS data sharing from IBM Data Server Drivers or Clients. In the multi-transport model, a commit may flow on a separate transport from the datasource connection. If you use an XA transaction manager that uses the multi-transport model, you still need to connect to DB2 for z/OS data sharing with DB2 Connect Server.

For more information about XA support in DB2 for z/OS, refer to 5.8, “XA transactions” on page 225.

Archived



## Part 2

# Setup and configuration

In this part we provide a description of the steps needed for the installation of a distributed environment.

This part contains the following chapters:

- ▶ Chapter 3, “Installation and configuration” on page 69
- ▶ Chapter 4, “Security” on page 129

Archived



## Installation and configuration

In this chapter we provide information about the steps to enable your DB2 for z/OS for distributed data access in a TCP/IP environment. We provide general configuration information about the DB2 system, as well as recommendations for other system components like TCP/IP, UNIX System Services, WLM, DB2 Connect, and the Data Server Drivers.

This chapter contains the following sections:

- ▶ “TCP/IP setup” on page 70
- ▶ “DB2 system configuration” on page 85
- ▶ “Workload Manager setup” on page 105
- ▶ “DB2 for LUW to DB2 for z/OS setup” on page 114
- ▶ “DRDA sample setup—From DB2 for z/OS requester to DB2 for LUW on AIX server” on page 122
- ▶ “Character conversion: Unicode” on page 125
- ▶ “Restrictions on the use of local datetime formats” on page 126
- ▶ “HiperSockets: Definition” on page 127

## 3.1 TCP/IP setup

Before DB2 can participate as a DRDA Application Server (AS) or DRDA Application Requester (AR) in a TCP/IP environment, you must set up TCP/IP for DB2. In this section we highlight the steps required for UNIX System Services, Language Environment® support and basic TCP/IP setup. We then describe the process for and give examples of defining TCP/IP to support data sharing, dynamic virtual I/P addressing (DVIPA) and LOCATION ALIAS support.

Refer to informational APAR II14203 for latest recommended maintenance for DB2 Version 9.1 for z/OS relating to DDF functions.

### 3.1.1 UNIX System Services setup

DDF uses the UNIX System Services assembler callable interface for TCP/IP services. Some of the functions that the DDF address space performs require that the user ID associated with the ssidDIST address space be a superuser. A superuser has an OMVS user ID value of UID(0). To see whether your DDF is already defined as a superuser, find the OMVS user ID of the ssidDIST address space.

From System Display and Search Facility (SDSF), use the Display Active panel to find the DDF address space output. You should see information similar to what we show in Figure 3-1.

```
IEF695I START D9C1DIST WITH JOBNAME D9C1DIST IS ASSIGNED TO USER STC      , GROUP SYS1
```

Figure 3-1 Output of D9C1DIST started task as seen from SDSF

The 'STC' after 'USER' indicates that D9C1DIST is assigned an OMVS user ID of STC. You can then use the RACF® command shown in Example 3-1, substituting STC for user ID to learn whether DDF is a superuser.

*Example 3-1 Displaying an OMVS user*

```
LISTUSER userid OMVS
```

The output of the command is shown in Figure 3-2.

```
OMVS INFORMATION
-----
UID= 0000000000
HOME= /u/stc
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMA= NONE
PROCUSERMA= NONE
THREADSMA= NONE
MMAPAREAMA= NONE
```

Figure 3-2 Output of LISTUSER STC OMVS command

This display indicates that OMVS user STC has a UID(0), so DDF is a superuser. If you need to define the ssidDIST user ID (distuid) as a superuser, you can use one of the RACF commands shown in Example 3-2.

*Example 3-2 Defining a superuser*

---

```
ADDUSER distuid OMVS(UID(0))... (If you are adding the user first time)
ALTUSER distuid OMVS(UID(0))... (If you are altering the existing user)
```

---

Refer to the *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846, and the *z/OS V1R10.0 Security Server RACF System Programmer's Guide*, SA22-7681, for more information about this topic.

### 3.1.2 Language Environment considerations

DDF uses some of the functions provided by Language Environment, so DB2 needs access to the Language Environment runtime library. There are two ways to achieve this:

- ▶ Include the Language Environment runtime library in the STEPLIB concatenation of the DDF address space startup. In this case, the Language Environment runtime library must be APF authorized. The DB2 installation automatically adds the library to the DDF STEPLIB concatenation.
- ▶ Concatenate the Language Environment library to the z/OS link list. In this case, the library does not need to be APF authorized. If you choose this approach, remove the Language Environment library concatenation from the DDF JCL procedure.

### 3.1.3 Basic TCP/IP setup

In this section we review the TCP/IP concepts and how they relate to DB2 as a DRDA AR and DRDA AS. We also discuss changing the TCP/IP files to support the requirements of your DB2 subsystem.

#### TCP/IP concepts and terminology

DB2 interacts with distributed clients and servers using the distributed data facility (DDF), which executes in the ssidDIST address space. DDF performs both DRDA AS functions, when DB2 is the data server, and DRDA AR functions, when DB2 is a requester. It is important to differentiate these roles when determining what TCP/IP specifications are necessary to support your requirements. Before we address these roles, we provide the following brief definitions of TCP/IP terms, which should be useful to you as you read this section.

- ▶ IP address

Uniquely identifies a host within the TCP/IP network. This is sometimes called an internet address. A DB2 subsystem resides on a TCP/IP host.

DB2 for z/OS supports both IP Version 4 addresses (which look like dotted decimal, e.g. 1.2.3.4) and IP Version 6 addresses (which are colon delimited, e.g. 2001:0DB8:0000:0000:0008:0800:200C:417A, which can also be abbreviated as 2001:DB8::8:800:200C:417A).

DB2 9 for z/OS is an IPv6 system and it displays all addresses (IPv4 or IPv6) in IPv6 colonhex format.

- ▶ Virtual IP Address (VIPA)

A VIPA is a generic term that refers to an internet address on a z/OS host that is not associated with a physical adapter. There are two types of VIPAs, static and dynamic (DVIPA). A static VIPA cannot be changed except through a VARY TCPIP,,OBEYFILE operator command.

- ▶ Dynamic VIPA (DVIPA)

A DVIPA can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility. Dynamic VIPAs are used to implement Sysplex Distributor.

- ▶ Distributed DVIPA

A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a sysplex. In DB2 data sharing, the group DVIPA, which is used to represent the group, is a distributed DVIPA.

- ▶ Domain name

The fully qualified name that identifies an IP address. This can be used instead of the IP address. An example of a domain name is stlmvs1.stl.ibm.com.

- ▶ Domain name server (DNS)

A DNS manages a distributed directory of domain names and related IP addresses. Domain names can be translated into IP addresses and you can find a domain name associated with a given IP address.

- ▶ Port

A port identifies an application executing in a host. For example, a port number identifies a DB2 subsystem to TCP/IP. There are three basic kinds of TCP/IP ports:

- Well-known port

This is a port number between 1 and 1023 that is reserved in the TCP/IP architecture for a specific TCP/IP application. Some typical well-known port numbers are: FTP (port number 21), Telnet (port number 23), and DRDA relational database (port number 446).

- Ephemeral port

Port numbers that are dynamically assigned to a client process by the client's TCP/IP instance. DB2 uses an ephemeral port when it is acting as the DRDA application requester (AR). This ephemeral port is associated with the requester for the life of the thread or connection.

- Server port

Port numbers that are used when a TCP/IP program does not have a well-known port number, or another instance of the server program is already installed using the well-known port number. If two different DB2 subsystems reside on the same host, acting as two different locations (in other words, not members of the same data sharing group), each DB2 subsystem must have a unique port.

DB2 uses the TCP/IP ports for three purposes:

- SQL port

This is the well-known port or the server port and is what most requesters will use to specify a DB2 subsystem or a DB2 data sharing group. (A data sharing group has a single SQL port that each member specifies.)

- Resync port

The resynchronization port is used by DB2 to handle two-phase commit processing, including indoubts. In a data sharing group, each member has a unique resync port, or RESPORT.

- Secure port

Optional. The secure port (or SECPORT) is the port DB2 uses for TCP/IP Secure Socket Layer (SSL) support. There is one SECPORT for a data sharing group.

- Service name

Another way to refer to a port number. A network administrator can assign a service name for a remote location instead of using the port number.

TCP/IP uses domain names (or IP addresses) and port numbers (or service names) to uniquely identify DB2 subsystems in the TCP/IP network.

DB2 9 for z/OS supports both IPv4 (32-bit) and IPv6 (128-bit) addressing. To enable DB2 to support IPv6 traffic (either inbound or outbound), you must configure TCP/IP with the dual-mode stack. You enable TCP/IP dual mode stack by changing the BPXPRMxx PARMLIB member to add a NETWORK statement that includes DOMAINNAME(AF\_INET6). Refer to *z/OS V1R10 Communications Server IP: Configuration Guide*, SC31-8775 for more information about enabling IPv6 addresses with BPXPRMxx.

### ***DB2 as a data server***

No entry is required in the DB2 Communications Database CDB if your DB2 subsystem acts only as a data server in a TCP/IP environment.

As a server processing TCP/IP connection requests, DB2 uses the SQL port, either the well-known port, 446, or a server port. DB2 will use the resynchronization (resync) port for processing 2-phase commit resync requests. When DDF is started, the DB2 subsystem identifies itself to TCP/IP using the port specification in its bootstrap data set. The port specification includes the SQL port number, the resynchronization port number (RESPORT), and, optionally, the secure port number (SECPORT). Refer to 3.2.4, “Updating the BSDS” on page 97 for more information about the port specification.

Optionally, you can protect the port number(s) that DB2 uses from being used by any other task or job in the subsystem. You can do this with a port reservation entry in the TCP/IP profile data set, as in Example 3-3 on page 74.

DB2 9 also allows you to specify the IP address in the BSDS. Specifying the IP address in the BSDS allows DB2 to accept requests to its port on any IP address (INADDR\_ANY). This provides significant advantage in a data sharing environment over the DB2 V8 function. In DB2 9, if you specify the IP address in the BSDS you do not have to define a domain name to TCP/IP.

In DB2 for z/OS V8, the domain name must be defined to the TCP/IP host so that a DB2 subsystem can accept connections from remote locations.

### ***DB2 as a requester***

The domain name and the server port number (or service name) of the database server must be defined in the DB2 Communications Database (CDB) at a requesting DB2. If you use a port number in the CDB to access a remote DB2 location, the port number must be defined to TCP/IP. We show an example in 3.2.2, “Configuring the Communications Database” on page 86.

TCP/IP will assign the requesting DB2 an ephemeral port for the life of the thread or connection. You do not need to specify an ephemeral port.

## Customizing TCP/IP data sets or files

There are several TCP/IP data sets or files that you may need to customize:

- ▶ The PROFILE data set
- ▶ The HOSTS data set
- ▶ The SERVICES data set

Once these are customized, you may need to update the DB2 bootstrap data set (BSDS) and configure the rest of the DB2 subsystem for DDF.

### The PROFILE data set

If you do not know the high level qualifier (hlq), check the JCL for the TCP/IP address space using SDSF. Figure 3-3 shows the JCL for our system.

```
//TCP/IP      JOB MSGLEVEL=1                                STC29433
//STARTING EXEC TCP/IP
XXTCP/IP     PROC P1='CTTRACE(CTIEZB00)',TCP/PROF=TCP/PROF,TCPDATA=TCPDATA 00010000
XX*                                                  00020000
XXTCP/IP EXEC PGM=EZBTCP/IP,REGION=OM,TIME=1440,      00030000
XX  PARM=&P1                                          00040000
XX*STEPLIB   DD DSN=TCP/IP.SEZATCP,DISP=SHR          00050000
IEFC653I SUBSTITUTION JCL - PGM=EZBTCP/IP,REGION=OM,TIME=1440,PARM=CTTRACE(CTIEZB00)
XXSYS/PRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=137,BLKSIZE=0) 00060000
XXSYS/ERR   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=137,BLKSIZE=0) 00070000
XXSYS/ERROR DD SYSOUT=*                                00080000
XXALG/PRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136) 00090000
XXCFG/PRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136) 00100000
XXSYS/OUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136) 00110000
XXCEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=137,BLKSIZE=0) 00120000
XX/PROFILE  DD DSN=TCP.&SYSNAME..TCP/PARMS(&TCP/PROF), 00130000
XX          DISP=SHR,FREE=CLOSE                     00140000
IEFC653I SUBSTITUTION JCL - DSN=TCP.SC63.TCP/PARMS(TCP/PROF),DISP=SHR,FREE=CLOSE
XXSYSTCPD   DD DSN=TCP.&SYSNAME..TCP/PARMS(&TCPDATA),DISP=SHR 00150000
IEFC653I SUBSTITUTION JCL - DSN=TCP.SC63.TCP/PARMS(TCPDATA),DISP=SHR
XXSYS/ABEND DD SYSOUT=*                                00160000
```

Figure 3-3 JCL of TCP/IP job, from SDSF display, showing high level qualifier for TCP/IP

The PROFILE DD card shows the hlq is 'TCP'. That DD card uses symbolic substitution to indicate the profile data set our system uses, TCP.SC63.TCPPARMS(TCP/PROF). This data set contains the PORT statement where you can reserve the SQL ports, resync ports, and secure ports for your DB2 subsystems. Example 3-3 shows a sample entry in a profile member for an environment with two standalone DB2 subsystems, DB2C and DB2D.

#### Example 3-3 Port reservations for two DB2 subsystems

```
PORT 446 TCP DB2CDIST ; DRDA SQL port for DB2C
PORT 5020 TCP DB2CDIST ; Resync port for DB2C
PORT 5021 TCP DB2DDIST ; DRDA SQL port for DB2D
PORT 5022 TCP DB2DDIST ; Resync port for DB2D
```

Only one DB2 subsystem can use the well-known port, 446. The other uses port 5021. Each DB2 defines a unique resynchronization port, 5020 and 5022, respectively. In this example the started procedure names for the DDF address spaces are DB2CDIST and DB2DDIST.

In our environment, no ports had been reserved for DB2. We did not reserve ports until defining the ports for our data sharing members. Refer to 3.1.4, “TCP/IP settings in a data sharing environment” on page 76 and Example 3-4 on page 77 for our port reservation statements.

We recommend you reserve the port numbers that DB2 will use in the TCP profile data set. Even though it is not required, it is good practice to specify the port numbers DB2 uses in the TCP profile data set. This also prevents other applications from starting to use these ports before DDF is started.

**Note:** If you are defining TCP/IP entries for a DB2 data sharing group in a parallel sysplex environment, you should use Dynamic Virtual IP Addresses (DVIPA). Refer to 3.1.4, “TCP/IP settings in a data sharing environment” on page 76.

### ***The HOSTS data set***

If you are defining TCP/IP support for the first time, or adding new hosts for DB2 access, you must define the TCP/IP host names that DB2 needs to know. These include the local host name, which must be defined before DDF is started. All domain names referenced in the table SYSIBM.IPNAMES must be defined. (Refer to 3.2.2, “Configuring the Communications Database” on page 86).

Define the host names by configuring the *hlq*.HOSTS.LOCAL data set, the */etc./hosts* file in the hierarchical file system (HFS), or the domain name server (DNS). Once these are configured, execute the MAKESITE utility to generate the *hlq*.HOSTS.ADDRINFO and the *hlq*.HOSTS.SITEINFO data sets. Refer to *z/OS V1R10.0 Communications Server: IP System Administrator's Commands*, SC31-8781, for more information about the MAKESITE utility.

Figure 3-4 on page 76 shows the starting point of the TCP.HOSTS.LOCAL data set. The three LPARs we used for our scenarios are SC63, SC64 and SC70, with host IP addresses of 9.12.6.70, 9.12.6.9 and 9.12.4.202, respectively.

```

BROWSE    TCP.HOSTS.LOCAL
Command ==>
***** Top of Data *****
HOST : 9.12.6.70 : WTSC63 ::::
HOST : 9.12.6.70 : WTSC63.ITSO.IBM.COM ::::
HOST : 9.12.6.9 : WTSC64 ::::
HOST : 9.12.6.9 : WTSC64.ITSO.IBM.COM ::::
HOST : 9.12.4.48 : WTSC65 ::::
HOST : 9.12.4.48 : WTSC65.ITSO.IBM.COM ::::
HOST : 9.12.4.202 : WTSC70 ::::
HOST : 9.12.4.202 : WTSC70.ITSO.IBM.COM ::::
HOST : 9.12.6.71 : WTSC630E ::::
HOST : 9.12.6.71 : WTSC630E.ITSO.IBM.COM ::::
HOST : 9.12.6.31 : WTSC640E ::::
HOST : 9.12.6.31 : WTSC640E.ITSO.IBM.COM ::::
HOST : 9.12.4.49 : WTSC650E ::::
HOST : 9.12.4.49 : WTSC650E.ITSO.IBM.COM ::::
HOST : 9.12.4.203 : WTSC700E ::::
HOST : 9.12.4.203 : WTSC700E.ITSO.IBM.COM ::::
HOST : 9.12.8.106 : TWSCJSC ::::
HOST : 9.12.8.106 : TWSCJSC.ITSO.IBM.COM ::::

```

Figure 3-4 Starting point for TCP.HOSTS.LOCAL

Later we added definitions to /SC63/etc/hosts to support DVIPA. Refer to Figure 3-11 on page 81 for this example of /etc/hosts definitions.

### **The SERVICES data set**

You must define the TCP/IP service names that DB2 needs to know. Configure the hlq.ETC.SERVICES data set or the /etc/services file in the HFS. If service names are present in the CDB (in field PORT of table SYSIBM.LOCATIONS), they must be defined in the z/OS data set or the HFS. An example of hlq.ETC.SERVICES entry is as follows:

```
DRDA 446/tcp ; DRDA databases
```

### **Update the BSDS**

You must update the bootstrap data set (BSDS) to include the TCP/IP port numbers. Refer to 3.2.4, “Updating the BSDS” on page 97, where we discuss the updates and provide examples and displays.

If you are not operating in a parallel sysplex and DB2 data sharing environment, you can proceed to 3.2, “DB2 system configuration” on page 85. The remainder of this section relates to TCP/IP definitions to support various aspects of data sharing.

## **3.1.4 TCP/IP settings in a data sharing environment**

**Important:** For high availability and workload balancing in a DB2 data sharing environment you must define TCP/IP and DB2 to take advantage of the function of Sysplex Distributor to make an initial connection to an available member of the data sharing group and to define the DB2 data sharing group and members with dynamic virtual IP addressing (DVIPA) to allow connection requests to succeed despite LPAR outages.



In a high availability configuration the DB2 data sharing group has a distributed DVIPA (also known as the group DVIPA). The requesters in the network specify this group DVIPA to initiate a connection to the data sharing group. The Sysplex Distributor function of TCP/IP identifies an available member of the data sharing group. Sysplex distributor generally routes initial connection requests evenly across the available members of the group. Subsequently the workload balancing functions available in the DRDA requesters use the server list returned by DB2 and route transactions or connection requests across the members of the data sharing group. Refer to Chapter 6, “Data sharing” on page 233 for details on workload balancing and the server list.

In DB2 9 for z/OS, there are three sets of required definitions; the first two comprise PORT and VIPADYNAMIC statements in the TCP/IP PROFILE data set, the third defines DVIPA addresses in the DB2 BSDS. After we describe these definitions for DB2 9 we describe the DB2 for z/OS V8 definitions for high availability and workload balancing.

In the examples that follow, we use the DVIPA addresses that we defined during our project. For a more complete narrative of the steps we took, refer to 3.1.5, “Sample DB2 data sharing DVIPA and Sysplex Distributor setup” on page 79 and 3.2.4, “Updating the BSDS” on page 97.

## PORT statements: DB2 9 for z/OS

If you are using a DB2 data sharing group in a parallel sysplex, each member of the data sharing group uses the same SQL port to receive incoming requests. However, each member must have a unique RESYNC port.

TCP/IP normally does not allow multiple applications to use the same port number. In a data sharing environment, it is possible to start multiple DB2 members on the same z/OS system, either as normal operation or as a temporary measure after a system failure. If two members of the same DB2 data sharing group start on the same LPAR, they will try to use the same SQL port number. To allow multiple applications to share the same port, use the SHAREPORT keyword in the TCP/IP profile data set. Example 3-4 shows the port reservation statements for the three members of our data sharing group.

*Example 3-4 Port reservations for three members of a DB2 data sharing group*

---

<b>38320</b>	TCP	<b>D9C1DIST</b>	<b>SHAREPORT</b>	;	SQL PORT
38321	TCP	D9C1DIST		;	Resync PORT
<b>38320</b>	TCP	<b>D9C2DIST</b>	<b>SHAREPORT</b>	;	SQL PORT
38322	TCP	D9C2DIST		;	Resync PORT
<b>38320</b>	TCP	<b>D9C3DIST</b>	<b>SHAREPORT</b>	;	SQL PORT
38323	TCP	D9C3DIST		;	Resync PORT

---

The SHAREPORT keyword allows multiple listeners (in our case D9C1DIST, D9C2DIST and D9C3DIST) to listen on the same port, 38320. Make sure you make this change on all members of the sysplex. The resync ports are not shared.

## VIPADYNAMIC statements

To enable Sysplex Distributor function to distribute connection requests across the members of your data sharing group, you must define the distribution as part of the VipaDynamic section of the TCP PROFILE data set. Include VipaRange keywords in each LPAR’s TCP/IP PROFILE data set. Specify VipaDefine and VipaDistribute Define keywords in the LPAR where the SD function will reside. Then specify VipaBackup keywords for the other LPARs.

Figure 3-5 shows the VipaDynamic statements that specify the SD, VipaDefine and VipaDistribute Define, in the TCP PROFILE data set on the SC70 LPAR where our Sysplex Distributor resided.

```
VIPADYNAMIC
VIPARANGE DEFINE 255.255.255.255 9.12.4.103 ; D9C1
VIPARANGE DEFINE 255.255.255.255 9.12.4.104 ; D9C2
VIPARANGE DEFINE 255.255.255.255 9.12.4.105 ; D9C3
VIPADEFINE 255.255.255.255 9.12.4.102 ; Group DVIPA
VIPADISTRIBUTE DEFINE 9.12.4.102 PORT 38320 DESTIP ALL
ENDVIPADYNAMIC
```

Figure 3-5 VipaDynamic statements including Sysplex Distributor definition for SC70

Figure 3-6 shows the VipaDynamic statements for the TCP PROFILE data set for SC64, which indicates 'VIPABACKUP' to provide backup Sysplex Distributor function. If the TCP on SC70 failed, the TCP on SC64 could take over the Sysplex Distributor function. The value after VIPABACKUP, in this case 100, is a relative number.

```
VIPADYNAMIC
VIPARANGE DEFINE 255.255.255.255 9.12.4.103 ; D9C1
VIPARANGE DEFINE 255.255.255.255 9.12.4.104 ; D9C2
VIPARANGE DEFINE 255.255.255.255 9.12.4.105 ; D9C3
VIPABACKUP 100 9.12.4.102 ; Group DVIPA
ENDVIPADYNAMIC
```

Figure 3-6 VipaDynamic statements with backup SD for SC64

Figure 3-7 shows the VipaDynamic statements for the TCP PROFILE data set for SC63, where the VIPABACKUP value is 200. In our configuration, '100' for SC64 is less than the '200' for SC63, so SC64 would be the first backup for Sysplex Distributor function and SC63 would be the second.

```
VIPADYNAMIC
VIPARANGE DEFINE 255.255.255.255 9.12.4.103 ; D9C1
VIPARANGE DEFINE 255.255.255.255 9.12.4.104 ; D9C2
VIPARANGE DEFINE 255.255.255.255 9.12.4.105 ; D9C3
VIPABACKUP 200 9.12.4.102 ; Group DVIPA
ENDVIPADYNAMIC
```

Figure 3-7 VipaDynamic statements with backup SD for SC63

## BSDS and DVIPA addresses: DB2 9 for z/OS

With DB2 9 for z/OS, you can specify the group DVIPA and member-specific DVIPA in the DB2 member's BSDS. Figure 3-8 on page 79 shows the DSNJU003 input for member D9C1 of our data sharing group. The group DVIPA, which specifies the data sharing group, is 9.12.4.102. The member-specific DVIPA, which only refers to D9C1, is 9.12.4.103.

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
// DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF IPV4=9.12.4.103,GRPIPV4=9.12.4.102
```

Figure 3-8 *BSDS specification with member-specific DVIPA and group DVIPA.*

Make the corresponding changes for each DB2 member's BSDS. Remember, you must stop DB2 before running DSNJU003. Refer to 3.2.4, "Updating the BSDS" on page 97 for a more complete discussion of our steps in updating the BSDS.

### DB2 for z/OS V8, DVIPA and BIND SPECIFIC

For a DB2 for z/OS data sharing group you still need a group DVIPA and a member-specific DVIPA for each member. The member-specific DVIPA supports workload balancing, allows for the DB2 members to start on an available LPAR, and allows clients to reach the correct DB2 member in case of resynchronization. In DB2 for z/OS V8 you cannot specify these DVIPAs in the BSDS. Instead, you must use the port specification section of the PROFILE data set to bind each member to the group DVIPA and to a member-specific DVIPA. These statements need to be included in the TCP PROFILE data sets in each LPAR where the DB2 data sharing members may run.

Figure 3-9 shows an example of binding ports to specific IP addresses. This is what we would have defined had we been using DB2 for z/OS V8 during our project.

```
38320 TCP D9C1DIST SHAREPORT BIND 9.12.4.102 ; SQL PORT
38321 TCP D9C1DIST BIND 9.12.4.103 ; Resync PORT
38320 TCP D9C2DIST SHAREPORT BIND 9.12.4.102 ; SQL PORT
38322 TCP D9C2DIST BIND 9.12.4.104 ; Resync PORT
38320 TCP D9C3DIST SHAREPORT BIND 9.12.4.102 ; SQL PORT
38323 TCP D9C3DIST BIND 9.12.4.105 ; Resync PORT
```

Figure 3-9 *DB2 for z/OS V8: Port statements binding a specific IP address to the DB2 ports*

Using BIND SPECIFIC for DB2 for z/OS V8 provides the required support for high availability and workload balancing. The disadvantage of this approach is that DB2 cannot accept TCP/IP requests using INADDR\_ANY. Make sure you use the DB2 9 for z/OS capability to define the group and member DVIPAs in the BSDS when you migrate to DB2 9 for z/OS.

### 3.1.5 Sample DB2 data sharing DVIPA and Sysplex Distributor setup

As mentioned earlier, our beginning configuration did not include port reservation statements and, therefore, did not specify SHAREPORT. With only the SQL port and resync port specified in the BSDS, and no port reservation statement, the members of our data sharing group were able to listen on any IP address for requests to their ports.

Figure 3-10 shows the output, on LPAR SC63, of D TCPIP,,N,CONN, a request to TCP to display connections. D9C1 is the member of the data sharing group on this LPAR. The last two lines indicate that D9C1 is listening on the SQL port for the data sharing group, 38320, and on the resync port, 38321, for D9C1, but that there are no IP addresses specified.

```

RESPONSE=SC63
EZZ2500I NETSTAT CS V1R10 TCPIP 125
USER ID  CONN      LOCAL SOCKET      FOREIGN SOCKET      STATE
CA9S02   000118E5 0.0.0.0..6000      0.0.0.0..0          LISTEN
CBDQDISP 0000B7FF 0.0.0.0..51107     0.0.0.0..0          LISTEN
DB8ADIST 00000040 0.0.0.0..12345     0.0.0.0..0          LISTEN
DB8ADIST 0000004A 0.0.0.0..12346     0.0.0.0..0          LISTEN
DB9ADIST 00018D30 0.0.0.0..12347     0.0.0.0..0          LISTEN
DB9ADIST 00018D51 0.0.0.0..12348     0.0.0.0..0          LISTEN
DB9ADIST 0001A09B 9.12.6.70..12347    9.12.6.70..17965     ESTBLSH
DB9ADIST 00018FAB 9.12.6.70..12347    9.12.6.70..17948     ESTBLSH
DB9ADIST 00018D31 0.0.0.0..12349     0.0.0.0..0          LISTEN
DB9BDIST 00019F42 0.0.0.0..12351     0.0.0.0..0          LISTEN
DB9BDIST 00019F3D 0.0.0.0..12350     0.0.0.0..0          LISTEN
DFSKERN   0000002D 0.0.0.0..139       0.0.0.0..0          LISTEN
D8F1DIST 0000004C 0.0.0.0..38051     0.0.0.0..0          LISTEN
D8F1DIST 00000047 0.0.0.0..38050     0.0.0.0..0          LISTEN
D9C1DIST 000106FD 0.0.0.0..38321    0.0.0.0..0          LISTEN
D9C1DIST 000105BF 0.0.0.0..38320    0.0.0.0..0          LISTEN

```

Figure 3-10 Output of D TCPIP,,N,CONN command

This definition, with the DRDA and resync port numbers specified in the BSDS but no IP address, makes it easy to reach any single member of the data sharing group, assuming they are all active, but it does not meet best practice for high availability. Best practice for high availability would allow any member to be restarted on another LPAR, in case of an LPAR outage, and would allow any available member to accept an incoming request. With our beginning configuration, we would not have been able to start D9C1, for example, on SC64, because we did not have SHAREPORT specified. And if a request for had been made for the SQL port, 38320, at IP address 9.12.6.70, but SC63 were not running, the request would simply fail.

When we were ready to implement DVIPA and Sysplex Distributor support in our data sharing environment we had to take several steps. You can use these steps when you implement DVIPA and SD support in your data sharing environment.

- ▶ Identify available IP addresses for the group DVIPA and each member-specific DVIPA
- ▶ Add port reservations statements specifying SHAREPORT
- ▶ Add VipaDynamic statements
- ▶ Add group and member-specific DVIPAs to the etc/hosts file
- ▶ Update BSDS for each DB2 member

After we successfully accomplished these steps, we added LOCATION ALIAS support to our environment. In this section we explain the steps we took and provide examples and display output for the TCP related tasks. We document the BSDS updates in 3.2.4, “Updating the BSDS” on page 97.

**Note:** All of these examples below relate to a DB2 9 for z/OS environment where the IPV4 addresses for group DVIPA and member-specific DVIPA are defined in the BSDS.

### **Identify IP addresses for group DVIPA and member-specific DVIPA**

Contact your TCP/IP network administrator to identify what addresses are available for group DVIPA and member-specific DVIPAs. If you are planning for a new data sharing group, and if your existing DB2 requesters use an IP address that is specific to DB2, try to make that IP address the group DVIPA, as it will ease the migration.

As you may have noticed in the examples and figures in the preceding section, we received the following IP addresses for our three-way data sharing group:

- 9.12.4.102 Group DVIPA
- 9.12.4.103 Member-specific DVIPA for D9C1
- 9.12.4.104 Member-specific DVIPA for D9C2
- 9.12.4.105 Member-specific DVIPA for D9C3

### **Update TCP PROFILE data sets**

The next step is to update the TCP PROFILE data sets. If you have not already done so, reserve your DB2 ports. We repeat our port reservation statements in Example 3-5. Add the same set of statements to each TCP PROFILE data set for the LPARs where your DB2 data sharing members may run. We added this to each profile member for SC63, SC64, and SC70. This example is only valid for DB2 9 for z/OS environments where IPV4 and IPV6 addresses have been added to the BSDS.

*Example 3-5 Port reservation statements for our three-way data sharing group*

---

38320	TCP	D9C1DIST	SHAREPORT		; SQL PORT
38321	TCP	D9C1DIST			; Resync PORT
38320	TCP	D9C2DIST	SHAREPORT		; SQL PORT
38322	TCP	D9C2DIST			; Resync PORT
38320	TCP	D9C3DIST	SHAREPORT		; SQL PORT
38323	TCP	D9C3DIST			; Resync PORT

---

Add VipaDynamic statements in each LPAR. In this case, each is slightly different. The VIPARANGE DEFINE statements will be identical, but only one profile will specify the VIPADEFINE statement for the group DVIPA and the VIPADISTRIBUTE statement. Refer to Figure 3-5 on page 78 for the VIPADEFINE and VIPADISTRIBUTE statements we used.

The others will specify the VIPABACKUP statement with a unique numeric value. In our case, we used 200 for SC63 and 100 for SC64. Refer to Figure 3-6 on page 78 for our statements for SC64 and to Figure 3-7 on page 78 for our statements for SC63.

### **Update etc/hosts file**

The next step is to update the etc/hosts file with the domain name that corresponds to the group DVIPA and the member-specific DVIPAs. Figure 3-11 shows the new entries in the etc/hosts file for SC63.

9.12.6.70	wtsc63.itso.ibm.com	wtsc63
9.12.6.71	wtsc63oe.itso.ibm.com	wtsc63oe
127.0.0.1	localhost.localdomain	localhost
9.12.4.166	d8fg.itso.ibm.com	d8fg
9.12.4.102	d9cg.itso.ibm.com	d9cg
9.12.4.103	d9cg.itso.ibm.com	d9cg
9.12.4.104	d9cg.itso.ibm.com	d9cg
9.12.4.105	d9cg.itso.ibm.com	d9cg

*Figure 3-11 Contents of /SC63/etc/hosts including DVIPA addresses*

We then issued the following command:

```
D TCPIP,,NETSTAT,HOME
```

Figure 3-12 shows the output, including the new group DVIPA and member-specific DVIPA for D9C1. The 'P' flag shows the primary interface for TCP/IP, which is the original IP address for SC63. The 'I' shows the internally generated DVIPA, our group DVIPA.

```
RESPONSE=SC63
EZZ2500I NETSTAT CS V1R10 TCPIP 596
HOME ADDRESS LIST:
ADDRESS          LINK          FLG
9.12.6.70        OSA2000LNK      P
9.12.6.71        OSA2020LNK
10.1.1.2         HIPERLF1
10.1.101.63      EZASAMEMVS
10.1.101.63      IQDIOLNK0A01653F
9.12.4.102       VIPL090C0466    I
9.12.4.103       VIPL090C0467
127.0.0.1        LOOPBACK
8 OF 8 RECORDS DISPLAYED
END OF THE REPORT
```

Figure 3-12 Output of D TCPIP,,NETSTAT,HOME command

The last step is to update the BSDS. Refer to 3.2.4, “Updating the BSDS” on page 97 for our discussion of this step.

## Alias support

Beginning with DB2 for z/OS V8 you can use the LOCATION ALIAS function to specify a subset of a data sharing group. Use this function to restrict the members of your data sharing group to which a requester can connect.

We defined two LOCATION ALIASes for our data sharing group. The first one, called DB9CALIAS, we used to specify member D9C1. The second alias, called DB9CSUBSET, we used to specify members D9C1 and D9C2.

To specify alias support, update your port reservations statements to reserve the new aliases. We assigned port 38324 to DB9CALIAS and port 38325 to DB9CSUBSET. Refer to Example 3-6 for the statements we used. Note that a single port serves for all members that use a specific alias. SHAREPORT is not required for a single-member alias.

Example 3-6 Port reservations for three members including aliases

---

```
38320 TCP D9C1DIST SHAREPORT ; SQL PORT
38321 TCP D9C1DIST           ; Resync PORT
38320 TCP D9C2DIST SHAREPORT ; SQL PORT
38322 TCP D9C2DIST           ; Resync PORT
38320 TCP D9C3DIST SHAREPORT ; SQL PORT
38323 TCP D9C3DIST           ; Resync PORT
38324 TCP D9C1DIST           ; Alias PORT for DB9CALIAS
38325 TCP D9C1DIST SHAREPORT ; Alias PORT for DB9CSUBSET
38325 TCP D9C2DIST SHAREPORT ; Alias PORT for DB9CSUBSET
```

---

We had to update our BSDS to record the new alias specifications. Refer to 3.2.4, “Updating the BSDS” on page 97 for the details. When we completed our ALIAS specifications, we issued the following command:

```
D TCPIP,,N,CONN
```

Figure 3-13 shows the output for SC63. D9C1DIST is listening on the SQL port, 38320, the resync port, 38321, and the alias ports, 38324 and 38325.

```
RESPONSE=SC63
EZZ2500I NETSTAT CS V1R10 TCP/IP 858
USER ID  CONN      LOCAL SOCKET      FOREIGN SOCKET      STATE
CA9S02   0001CB4D 0.0.0.0..6000      0.0.0.0..0          LISTEN
CBDQDISP 0000B7FF 0.0.0.0..51107     0.0.0.0..0          LISTEN
DB8ADIST 00000040 0.0.0.0..12345     0.0.0.0..0          LISTEN
DB8ADIST 0000004A 0.0.0.0..12346     0.0.0.0..0          LISTEN
DB9ADIST 0001D872 9.12.6.70..12347   9.12.6.70..18016    ESTBLSH
DB9ADIST 0001CB3A 0.0.0.0..12347     0.0.0.0..0          LISTEN
DB9ADIST 0001CB4F 0.0.0.0..12348     0.0.0.0..0          LISTEN
DB9ADIST 0001CB3B 0.0.0.0..12349     0.0.0.0..0          LISTEN
DB9BDIST 00019F42 0.0.0.0..12351     0.0.0.0..0          LISTEN
DB9BDIST 00019F3D 0.0.0.0..12350     0.0.0.0..0          LISTEN
DFSKERN  0000002D 0.0.0.0..139       0.0.0.0..0          LISTEN
D8F1DIST 0000004C 0.0.0.0..38051     0.0.0.0..0          LISTEN
D8F1DIST 00000047 0.0.0.0..38050     0.0.0.0..0          LISTEN
D9C1DIST 0001B928 9.12.6.70..38320   9.12.5.149..48835   ESTBLSH
D9C1DIST 0001B922 9.12.6.70..38320   9.12.5.149..48828   ESTBLSH
D9C1DIST 0001B921 0.0.0.0..38321    0.0.0.0..0          LISTEN
D9C1DIST 0001B919 9.12.6.70..38320   9.12.5.149..48822   ESTBLSH
D9C1DIST 0001B910 0.0.0.0..38320    0.0.0.0..0          LISTEN
D9C1DIST 0001B915 0.0.0.0..38325    0.0.0.0..0          LISTEN
D9C1DIST 0001B91F 9.12.4.103..38320  9.12.5.149..48819   ESTBLSH
D9C1DIST 0001B913 0.0.0.0..38324    0.0.0.0..0          LISTEN
D9D1DIST 0000004B 0.0.0.0..38331     0.0.0.0..0          LISTEN
D9D1DIST 00000042 0.0.0.0..38330     0.0.0.0..0          LISTEN
```

Figure 3-13 Output of D TCPIP,,N,CONN command

### 3.1.6 Starting DDF with TCP/IP

Many of the TCP/IP definitions we have described for our DB2 environment are visible in the DB2 DISPLAY DDF output. Figure 3-14 on page 84 shows the output for DB9A, our standalone DB2 member, with APAR PK80474 applied<sup>1</sup>.

<sup>1</sup> With APAR PK80474, message DSNL086I is eliminated in non-data sharing and a new message DSNL089I is added for data sharing.

```

RESPONSE=SC63
DSNL080I  -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICCLU
DSNL083I  DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I  TCPPORT=12347 SECPORT=12349 RESPORT=12348 IPNAME=-NONE
DSNL085I  IPADDR=:9.12.6.70
DSNL086I  SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Figure 3-14 Example of DISPLAY DDF from DB9A standalone DB2

The LOCATION, DB9A, the SQL port, 12347, the SECPORT, 12349, the RESPORT, 12348, the host IP address, and the domain names are all clear.

Figure 3-15 shows the display output for member D9C1. This display reflects the beginning configuration. The IPADDR and the MEMBER IPADDR are the same, and no ALIAS is defined.

```

RESPONSE=SC63
DSNL080I  -D9C1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICCLU
DSNL083I  DB9C              USIBMSC.SCPD9C1  -NONE
DSNL084I  TCPPORT=38320 SECPORT=0      RESPORT=38321 IPNAME=-NONE
DSNL085I  IPADDR=:9.12.6.70
DSNL086I  SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL086I  RESYNC  DOMAIN=wtsc63.itso.ibm.com
DSNL089I  MEMBER IPADDR=:9.12.6.70
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Figure 3-15 Example of DISPLAY DDF from D9C1 data sharing member

Figure 3-16 shows the display output for member D9C2, on SC64.

```

RESPONSE=SC64
DSNL080I  -D9C2 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICCLU
DSNL083I  DB9C              USIBMSC.SCPD9C2  -NONE
DSNL084I  TCPPORT=38320 SECPORT=0      RESPORT=38322 IPNAME=-NONE
DSNL085I  IPADDR=:9.12.4.102
DSNL086I  SQL      DOMAIN=-NONE
DSNL086I  RESYNC  DOMAIN=-NONE
DSNL089I  MEMBER IPADDR=:9.12.4.104
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Figure 3-16 Example of DISPLAY DDF from D9C2 data sharing member



Figure 3-17 shows the display output for member D9C3 on SC70.

These displays reflect the changes to support DVIPA. Notice that the IPADDR in each case is the group DVIPA, 9.12.4.102. Each display shows a unique value in MEMBER IPADDR.

```
RESPONSE=SC70
DSNL080I  -D9C3 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DB9C              USIBMSC.SCPD9C3  -NONE
DSNL084I  TCPPORT=38320  SECPORT=0      RESPORT=38323  IPNAME=-NONE
DSNL085I  IPADDR=:9.12.4.102
DSNL086I  SQL      DOMAIN=-NONE
DSNL086I  RESYNC   DOMAIN=-NONE
DSNL089I  MEMBER IPADDR=:9.12.4.105
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Figure 3-17 Example of DISPLAY DDF from D9C3 data sharing member

Each of the displays above includes the VTAM LUNAME. Even though the VTAM support was not required for our scenarios, it had been previously defined and we left it there.

## 3.2 DB2 system configuration

If your TCP/IP, UNIX System Services, and Language Environment environments are already defined, just define the DB2 resources to support distributed access. These resources are as follows:

- ▶ Shared memory object
- ▶ Communications database (CDB)
- ▶ DSNZPARMs
- ▶ Bootstrap data set (BSDS)
- ▶ DDF address space
- ▶ Support for ODBC, JDBC, stored procedures, and so forth

### 3.2.1 Defining the shared memory object

Beginning with DB2 9 for z/OS, DB2 DDF takes advantage of shared memory to pass SQL and data rows between the DIST address space and the DBM1 address space. Shared memory is a type of virtual storage that was introduced in z/OS 1.5 that allows multiple address spaces to address common storage. This memory resides above the 2 GB bar. The shared memory object is created at DB2 startup and the DB2 address spaces for the subsystem (ssidDIST, ssidDBM1, ssidMSTR, and Utilities) are registered with z/OS to access the shared memory object.

To define the size of the shared memory, use the HVSHARE parameter of the IEASYSxx member in the parmlib concatenation. Ensure that you have defined a high enough value for HVSHARE to satisfy all component requests for shared memory within the z/OS image. If you do not specify HVSHARE in your IEASYSxx member, the shared memory object will be created with the default value. The default value is 510 TB.

Use the following z/OS command to see what the current definition is and what the current allocation is: DISPLAY VIRTSTOR,HVSHARE.

Figure 3-18 shows that HVSHARE is defined at the default value of 510 TB on our system.

2009110	19:32:30.08	PAOLOR6	00000210	DISPLAY VIRTSTOR,HVSHARE
2009110	19:32:30.10	PAOLOR6	00000010	IAR019I 19.32.30 DISPLAY VIRTSTOR 369
		369	00000010	SOURCE = DEFAULT
		369	00000010	TOTAL SHARED = 522240G
		369	00000010	SHARED RANGE = 2048G-524288G
		369	00000010	SHARED ALLOCATED = 393217M

Figure 3-18 Results of DISPLAY VIRTSTOR,HVSHARE showing default definition

For more information about shared memory, refer to *z/OS V1R10.0 MVS Initialization and Tuning Reference*, SA22-7592.

### 3.2.2 Configuring the Communications Database

The Communications Database (CDB) is part of the DB2 Catalog and consists of a set of tables that DDF uses to establish communications with remote databases. DDF uses the CDB when providing DRDA AR functions. DDF does not use the CDB when performing DRDA AS functions in a TCP/IP environment.

If your DB2 for z/OS provides DRDA AR functions to request data from other systems, you must have one row in the SYSIBM.LOCATIONS table for each remote system you want to access. You also need a row in the SYSIBM.IPNAMES table for each remote system.

The following section provides a brief description of what each table contains and an example of the CDB tables as they were defined in our system.

#### **SYSIBM.LOCATIONS table**

DDF checks the SYSIBM.LOCATIONS table first when you issue a request to another system. DDF uses the LOCATIONS table to determine the port number or service name used to connect to the remote location. The column LINKNAME maps to the corresponding row in table IPNAMES.

The following LOCATIONS columns apply to requests to remote systems using TCP/IP:

##### ► LOCATION

This column identifies the remote system. You must provide a value for each system from which you intend to request data. This value is used on the SQL CONNECT TO statement, or as the location name in remote binds or three-part names.

##### ► LINKNAME

This column identifies the TCP/IP attributes for the location. For each link name, you must have a corresponding value in IPNAMES.

##### ► PORT

This column determines the port number to be used for the remote location. If blank, the default port number 446 is used. If the value is not blank, it is either the SQL port number of the remote system, or a TCP/IP service name, which can be converted to a TCP/IP port.

##### ► DBALIAS

Database alias. The name associated with the remote server. If DBALIAS is blank, the location name is used to access the remote database server. If DBALIAS is not blank and the name of any database object contains the location qualifier (in other words it is a

three-part name), this column does not change that name when the SQL is sent to the remote site.

► **TRUSTED**

This column indicates whether the connection to the remote server can be trusted.

► **SECURE**

This column indicates whether a secure connection using the Secure Socket Layer (SSL) protocol is required for outbound DRDA connections

Table 3-1 shows the values in SYSIBM.LOCATIONS during our project. In our environment the last three columns were blank, so we do not include them in the table.

*Table 3-1 SYSIBM.LOCATIONS*

LOCATION	LINKNAME	PORT
DB8A	DB8A	12345
DB9C	DB9C	38320
DB9C2	DB9C2	38320
KODIAK	KODIAK	50002
SAMPLE	MYUDBLNK	50002

### **SYSIBM.IPNAMES table**

The SYSIBM.IPNAMES table defines the remote DRDA servers DB2 can access using TCP/IP. The relevant columns are:

► **LINKNAME**

This column corresponds to the LINKNAME column of the LOCATIONS table. The LINKNAME column value must be unique within the IPNAMES table.

► **SECURITY\_OUT**

This column defines the DRDA security option that is used when local DB2 SQL applications connect to any remote server associated with the TCP/IP host specified by this LINKNAME:

– **A**

“already verified”, the default. Outbound connection requests contain an unencrypted authorization ID and no password. The authorization ID is either the DB2 user’s authorization ID or a translated ID, depending upon the value of the USERNAMES column.

– **D**

“user ID and security-sensitive data encryption”. Outbound connection requests contain an authorization ID and no password. The authorization ID is either the DB2 user’s authorization ID or a translated ID, depending upon the value of the USERNAMES column.

– **E**

“userid, password, and security-sensitive data encryption”. Outbound connection requests contain an authorization ID and a password. The password is obtained from the SYSIBM.USERNAMES table. The USERNAMES column must specify ‘O’.

– P

“password”. Outbound connection requests contain an authorization ID and a password. The password is obtained from the SYSIBM.USERNAMES table. The USERNAMES column must specify ‘O’. This option indicates that the user ID and the password are to be encrypted if cryptographic services are available at the requester and if the server supports encryption. Otherwise, the user ID and the password are sent to the partner in clear text.

– R

“RACF PassTicket”. Outbound connection requests contain a user ID and a RACF PassTicket. The value specified in the LINKNAME column is used as the RACF PassTicket application name for the remote server. The authorization ID used for an outbound request is either the DB2 user’s authorization ID or a translated ID, depending upon the value of the USERNAMES column. The authorization ID is not encrypted when it is sent to the partner.

► USERNAMES

This column controls outbound authorization ID transaction. In this column, you can specify ‘O’, ‘S’ or blank. If either ‘O’ or ‘S’ is specified, you must populate the SYSIBM.USERNAMES table.

– O

Specify ‘O’ if you want outbound translation. If the value in the security\_out column is ‘P’, then this column must be ‘O’.

– S

‘S’ indicates that a row in the SYSIBM.USERNAMES table is used to obtain the system AUTHID used to establish a trusted connection.

– blank

No translation occurs. Only outbound translation is supported with TCP/IP.

► IPADDR

This column contains an IPv4 or IPv6 address, or domain name of a remote TCP/IP host.

Table 3-2 shows the values in SYSIBM.IPNAMES during our project.

Table 3-2 SYSIBM.IPNAMES

LINKNAME	SECURITY_OUT	USERNAMES	IPADDR
DB8A	P	O	WTSC63.ITSO.IBM.COM
DB9C	A		WTSC64.ITSO.IBM.COM
DB9C2	P	O	WTSC63.ITSO.IBM.COM
KODIAK	P	O	KODIAK.ITSO.IBM.COM
MYUDBLNK	P	O	KODIAK.ITSO.IBM.COM

## SYSIBM.USERNAMES table

The USERNAMES table is used for outbound ID translation (TCP/IP and SNA) and inbound ID translation and 'come from' checking (SNA only).

### ► TYPE

Values indicate how DDF should use this row:

- I (SNA only)  
Inbound and 'come from' checking.
- O  
For outbound authorization ID translation.
- S  
For outbound system AUTHID to establish a trusted connection.

### ► AUTHID

Authorization ID to be translated. If blank, the translation applies to every authorization ID.

### ► LINKNAME

Relates the row in IPNAMES with the same LINKNAME to the values in this row to determine authorization ID translation.

### ► NEWAUTHID

Translated value of AUTHID. Blank specifies no translation. NEWAUTHID can be stored as encrypted data by calling the DSNLEUSR stored procedure. To send the encrypted value of AUTHID across a network, one of the encryption security options in the SYSIBM.IPNAMES table should be specified.

### ► PASSWORD

Password to accompany an outbound request, if passwords are not encrypted by RACF. If passwords are encrypted, or the row is for inbound requests, the column is not used. PASSWORD can be stored as encrypted data by calling the DSNLEUSR stored procedure. To send the encrypted value of PASSWORD across a network, one of the encryption security options in the SYSIBM.IPNAMES table should be specified.

Table 3-3 shows the values in SYSIBM.USERNAMES during our project.

Table 3-3 *SYSIBM.USERNAMES*

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O		DB8A	PAOLOR5	PUP4SALE
O		DB9C	PAOLOR4	123ABC
O		KODIAK	db2inst3	db2inst3
O		MYUDBLNK	DB2INST3	DB2INST3

## SYSIBM.IPLIST table

The IPLIST table allows multiple IP addresses to be specified for a given LOCATION. The same value for the IPADDR column cannot appear in both the IPNAMES table and the IPLIST table. Use of this table allows DDF to select a member of a data sharing group from this list for initial connection requests.

We do not recommend use of this table. Rather, indicate the IPADDR in the IPNAMES table and implement Sysplex Distributor support to allow any available member of a data sharing group to support the initial connection request. If you intend to limit the members of a data sharing group to which a DB2 can connect, use LOCATION ALIAS support. Refer to “Alias support” on page 82 and 3.2.4, “Updating the BSDS” on page 97 for examples of how we specified LOCATION ALIAS in our environment. For further discussion of using LOCATION ALIAS, refer to 6.2.1, “DB2 data sharing subsetting” on page 248.

We did not populate this table for our project.

## Changing the CDB

You can make changes to the CDB while DDF is active. Depending on the type of changes you make, these changes take effect at different times:

- ▶ Changes to USERNAMES take effect at the next thread access.
- ▶ If DDF has not yet started communicating to a particular location, IPNAMES and LOCATIONS take effect when DDF attempts to communicate with that location.
- ▶ If DDF has already started communication, changes to IPNAMES and LOCATIONS take effect the next time DDF is started.

## 3.2.3 DB2 installation parameters (DSNZPARM)

In this section we describe the DSNZPARM values that apply to DDF and distributed traffic. In addition we describe fields on the DB2 installation panels that are part of defining DDF but are not DSNZPARM entries. For further information about any of these parameters or fields, refer to *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846.

Table 3-4 provides a reference for the parameters, the panels on which they appear and their possible and default values. We discuss these parameters in the order we list them here.

Table 3-4 DSNZPARM parameters

PARAMETER	PANELID	Possible values	Default
MAXDBAT	DSNTIPE	0-1999	200
CONDBAT	DSNTIPE	0-150,000	10,000
DDF	DSNTIPR	NO, AUTO, COMMAND	NO
RLFERRD	DSNTIPR	NOLIMIT, NORUN, 1 to 5,000,000	NOLIMIT
RESYNC	DSNTIPR	1 to 99	2
CMTSTAT	DSNTIPR	ACTIVE, INACTIVE	INACTIVE
MAXTYPE1	DSNTIPR	0 to value of CONDBAT	0
IDTHTOIN	DSNTIPR	0 to 9999	120
EXTSEC	DSNTIPR	YES, NO	YES
TCPALVER	DSNTIP5	YES, NO	NO
EXTRAREQ	DSNTIP5	0 to 100	100
EXTRASRV	DSNTIP5	0 to 100	100
HOPAUTH	DSNTIP5	BOTH or RUNNER	BOTH
TCPKPALV	DSNTIP5	ENABLE, DISABLE, or 1 to 65534	120

PARAMETER	PANELID	Possible values	Default
POOLINAC	DSNTIP5	0 to 9999	120
ACCUMACC	DSNTIPN	NO, 2-65535	10
ACCUMUID	DSNTIPN	0-17	0
PRGSTRIN	none - PK46079	ENABLE, DISABLE	ENABLE
SQLINTRP	none - PK59385	ENABLE, DISABLE	ENABLE

The first installation panel that affects DDF is DSNTIPE, shown in Figure 3-19. This is where you specify how many concurrent distributed threads and distributed connections DDF can support.

```

DSNTIPE          INSTALL DB2 - THREAD MANAGEMENT
====>

Check numbers and reenter to change:
 1 DATABASES          ==> 100      Concurrently in use
 2 MAX USERS          ==> 200      Concurrently running in DB2
 3 MAX REMOTE ACTIVE   ==> 200      Maximum number of active
                                     database access threads
 4 MAX REMOTE CONNECTED ==> 10000   Maximum number of remote DDF
                                     connections that are supported
 5 MAX TSO CONNECT     ==> 50       Users on QMF or in DSN command
 6 MAX BATCH CONNECT   ==> 50       Users in DSN command or utilities
 7 SEQUENTIAL CACHE    ==> BYPASS   3990 storage for sequential IO.
                                     Values are SEQ or BYPASS.
 8 MAX KEPT DYN STMTS  ==> 5000     Maximum number of prepared dynamic
                                     statements saved past commit points
 9 CONTRACT THREAD STG ==> NO        Periodically free unused thread stg
10 MANAGE THREAD STORAGE ==> YES     Manage thread stg to minimize size
11 LONG-RUNNING READER ==> 0        Minutes before read claim warning
12 PAD INDEXES BY DEFAULT ==> NO     Pad new indexes by default
13 MAX OPEN FILE REFS  ==> 100      Maximum concurrent open data sets

PRESS: ENTER to continue  RETURN to exit  HELP for more information

```

Figure 3-19 DSNTIPE panel specifying MAXDBAT and CONDBAT values

► Max remote active (MAXDBAT)

Use this field to specify the maximum number of database access threads (DBATs) that can be active concurrently. The default is 200, and you can specify up to 1999, with the condition that the sum of MAXDBAT and CTHREAD (Field #2, Max Users, on DSNTIPE) cannot exceed 2000.

If requests for DBATs exceed MAXDBAT the allocation is allowed, but subsequent processing depends on whether you specified ACTIVE or INACTIVE in the DDF threads field (CMTSTAT) on panel DSNTIPR.

- If you specified INACTIVE (the default) the request will be processed when DB2 can assign an unused DBAT to the connection.
- If you specified ACTIVE, further processing is queued waiting for an active DBAT to terminate.

We recommend INACTIVE for most environments.

If you set MAXDBAT to zero, you prevent DB2 from accepting new distributed requests.

**Note:** The number of allied threads (CTHREAD) and DBATs that your DB2 system can handle concurrently is workload dependent. If you specify too high a sum (CTHREAD + MAXDBAT) and you experience a workload spike, you may exhaust the available virtual storage in the DBM1 address space. This could lead to a dramatic slowdown in processing as DB2 performs a system contraction. You may also cause DB2 to abend if you specify too large a number of concurrent threads. Evaluate your workload requirements for thread storage and set MAXDBAT and CTHREAD conservatively to ensure you do not encounter either of these conditions.

► Max remote connected (CONDBAT)

Use this field to specify the maximum number of concurrent remote connections. This value must be greater than or equal to MAXDBAT. If a request to allocate a new connection to DB2 is received and you have already reached CONDBAT, the connection request is rejected.

Set CONDBAT to be significantly higher than MAXDBAT. In general, a connection request should not be rejected. The cost of maintaining a connection while waiting for a thread is low.

Figure 3-20 on page 93 shows panel DSNTIPR, the first of the two panels for defining DDF. We discuss each field briefly before proceeding to DSNTIP5, the second DDF panel. The fields shown represent our initial input settings, not necessarily the defaults.



```

DSNTIPR          INSTALL DB2 - DISTRIBUTED DATA FACILITY
====>

Enter data below:

 1 DDF STARTUP OPTION  ===> AUTO      NO, AUTO, or COMMAND
 2 DB2 LOCATION NAME   ===> DB9A      The name other DB2s use to
                                         refer to this DB2
 3 DB2 NETWORK LUNAME  ===> SCPDB9A   The name VTAM uses to refer to this DB2
 4 DB2 NETWORK PASSWORD ===>          Password for DB2's VTAM application
 5 RLST ACCESS ERROR   ===> NOLIMIT   NOLIMIT, NORUN, or 1-5000000
 6 RESYNC INTERVAL     ===> 2        Minutes between resynchronization period
 7 DDF THREADS         ===> INACTIVE  Status of a qualifying database access
                                         thread after commit. ACTIVE or INACTIVE.
 8 MAX INACTIVE DBATS  ===> 0        Max inactive database activity threads
 9 DB2 GENERIC LUNAME  ===>          Generic VTAM LU name for this DB2
                                         subsystem or data sharing group
10 IDLE THREAD TIMEOUT ===> 120      0 or seconds until dormant server ACTIVE
                                         thread will be terminated (0-9999).
11 EXTENDED SECURITY   ===> YES      Allow change password and descriptive
                                         security error codes. YES or NO.

PRESS:  ENTER to continue  RETURN to exit  HELP for more information

```

Figure 3-20 DSNTIPR panel showing DDF values for subsystem DB9A

For each of the fields in this panel and the next panel, we repeat the field name, then indicate the DSNZPARM parameter name in parentheses. (none) means the field does not correspond to a DSNZPARM parameter, but is instead specified in the bootstrap data set (BSDS), which we discuss in 3.2.4, “Updating the BSDS” on page 97.

► DDF startup option (DDF)

Specify AUTO to have DDF initialized and started when DB2 starts. COMMAND also initializes DDF, but you must then enter the START DDF command. NO indicates you do not want the DDF active in this DB2.

► DB2 LOCATION name (none)

This is the unique name that identifies this DB2, or the data sharing group of which this DB2 is a member. Any requester specifies the LOCATION name and either an IP address or a service name to reach this DB2. Default is LOC1.

In a data sharing environment requesters can specify a LOCATION ALIAS. For a description of how to define a LOCATION ALIAS for DB2, refer to “Alias support” on page 82. For a discussion of using LOCATION ALIAS, refer to “6.2.1, “DB2 data sharing subsetting” on page 248.

► DB2 network LUNAME (none)

This name uniquely identifies DB2 to VTAM. This field is required for this panel even if you do not use VTAM in your environment. Default is LU1.

► DB2 network password (none)

This optional field specifies the password that VTAM uses to recognize this DB2 subsystem. We recommend you do not use this field, even if you use VTAM.

► RLST access error (RLFERRD)

For dynamic SQL, use this to specify what action DB2 takes if the governor cannot access the resource limit specification table, or if DB2 cannot find a row in the table for the query user.

- NOLIMIT means the dynamic SQL runs without limit.
- NORUN terminates dynamic SQL immediately with an SQL error code.
- A number from 1 to 5,000,000 indicates the default limit; if the limit is exceeded the dynamic SQL statement is terminated.

Refer to Example 7.4.3 on page 310 for further discussion of this topic.

► Resync interval (RESYNC)

This is the interval, in minutes, between resynchronization periods. DB2 processes indoubt logical units of work during these periods.

► DDF threads (CMTSTAT)

Use this field to indicate how DB2 treats a DBAT after a successful commit or rollback operation, when the DBAT holds no cursors.

If you specify ACTIVE, the DBAT remains active and continues to consume system resources. This restricts the number of connections you can support. If you must support a large number of connections, specify INACTIVE, which is the default.

If you specify INACTIVE, DB2 will use one of two concepts:

- For most cases, if the DBAT holds no cursors, has no declared global temporary tables defined, and has not specified KEEP DYNAMIC YES, then DB2 will disassociate the DBAT from the connection, mark the connection inactive, and return the DBAT to a pool for use by another connection. This concept is also called inactive connection. Because the DBAT is pooled, inactive connection support is more efficient in supporting a large number of connections.
- The second concept typically involves private protocol threads. In this case the thread remains associated with the connection, but thread storage is reduced. This concept is also called inactive DBAT, and was previously called a type 1 inactive thread. See the following bullet, “Max inactive DBATs (MAXTYPE1)”.

When a thread issues a COMMIT or ROLLBACK, DB2 tries to make it an inactive connection, and if that is not possible, DB2 tries to make it an inactive DBAT. If neither is possible, the thread remains active, potentially consuming valuable resources.

► Max inactive DBATs (MAXTYPE1)

This field limits the number of inactive DBATs in your system. If you choose the default, zero (0), then inactive DBATS are not allowed. If a thread would otherwise meet the requirements to become an inactive DBAT, it remains active. If you choose a value greater than zero, that becomes the high water mark for inactive DBATs. If a thread meets the requirements of an inactive DBAT, but max inactive DBATs is reached, the remote connection is terminated.

► DB2 generic LUNAME (none)

This applies only to a DB2 data sharing member using SNA to accept distributed traffic.

► Idle thread timeout (IDHTOIN)

This specifies the approximate time, in seconds, that an active server thread should be allowed to remain idle before it is canceled. After the timeout expires, the thread is canceled and its locks and cursors are released. Inactive and indoubt threads are not subject to IDHTOIN.

If you specify 0, you disable time-out processing. In this case, idle threads remain in the system and continue to hold resources. We recommend you choose a non-zero value.

► Extended security (EXTSEC)

This field specifies two security options with one variable: whether detailed reason codes are returned when a DDF connection request fails due to a security error, and whether RACF users can change their passwords. YES is the default and is recommended. NO returns generic codes and prevents RACF users from changing their passwords.

Figure 3-21 shows the second DDF panel, DSNTIP5. The fields shown represent our initial input settings, not necessarily the defaults.

```

DSNTIP5          INSTALL DB2 - DISTRIBUTED DATA FACILITY PANEL 2
====>

Enter data below:
 1 DRDA PORT          ====> 12347 TCP/IP port number for DRDA clients.
                           1-65534 (446 is reserved for DRDA)
 2 SECURE PORT        ====> 12349 TCP/IP port number for secure DRDA
                           clients. 1-65534 (448 is reserved
                           for DRDA using SSL)
 3 RESYNC PORT        ====> 12348 TCP/IP port for 2-phase commit. 1-65534
 4 TCP/IP ALREADY VERIFIED ====> NO Accept requests containing only a
                                     userid (no password)? YES or NO
 5 EXTRA BLOCKS REQ   ====> 100 Maximum extra query blocks when DB2 acts
                                     as a requester. 0-100
 6 EXTRA BLOCKS SRV   ====> 100 Maximum extra query blocks when DB2 acts
                                     as a server. 0-100
 7 AUTH AT HOP SITE    ====> BOTH Authorization at hop site. BOTH or RUNNER
 8 TCP/IP KEEPALIVE    ====> 120 ENABLE, DISABLE, or 1-65534
 9 POOL THREAD TIMEOUT ====> 120 0-9999 seconds

PRESS: ENTER to continue RETURN to exit HELP for more information

```

Figure 3-21 DSNTIP5 panel showing values for subsystem DB9A

► DRDA port (none)

Specify the TCP/IP port number used to accept TCP/IP requests from remote DRDA clients. This field is input to the DSNJU003 job that DB2 generates.

Remember to specify the same DRDA port on all members of a DB2 data sharing group.

► Secure port (none)

Specify this field if you intend to accept secure TCP/IP connection requests from remote DRDA clients. You must specify a value if you plan to use TCP/IP with Secure Socket Layer (SSL). This field is input to the DSNJU003 job that DB2 generates. Remember to specify the same secure port on all members of a DB2 data sharing group.

Refer to 4.3.3, “Secure Socket Layer” on page 151 for more information about SSL.

► Resync port (none)

This field is the TCP/IP port number that is used to process requests for two-phase commit resynchronization. This value must be different than the value that is specified for DRDA port.

In a data sharing environment, each member must have a unique resync port.

► TCP/IP already verified (TCPALVER)

Use this field to specify whether DB2 will accept TCP/IP connection requests that contain only a user ID, but no password, RACF PassTicket or Kerberos ticket. This value must be the same for all members of a data sharing group. This option applies to all incoming requests that use TCP/IP regardless of the requesting location.

We recommend the default, NO, which requires all requesting locations to use passwords, RACF PassTickets or Kerberos tickets when they send user IDs.

► Extra blocks req (EXTRAREQ)

This field specifies an upper limit on the number of extra DRDA query blocks DB2 requests from a remote DRDA server. This does not limit the size of the SQL query answer set; it simply controls the total amount of data that can be transmitted on any given network exchange. We recommend the default.

► Extra blocks srv (EXTRASRV)

This field specifies an upper limit on the number of extra DRDA query blocks that DB2 returns to a DRDA client. This does not limit the size of the SQL query answer set; it simply controls the total amount of data that can be transmitted on any given network exchange. We recommend taking the default.

► Auth at hop site (HOPAUTH)

This field indicates whose authorization is to be checked at a second server (also called a *hop* site) when the request is from a requester that is not DB2 for z/OS. This option applies only when private protocol access is used for the hop from the second to third site.

► TCP/IP keepalive (TCPKPALV)

You can use this field as an override in cases where the TCP/IP KeepAlive value in the TCP/IP configuration is not appropriate for the DB2 subsystem. The settings have the following meanings:

- ENABLE: Do not override the TCP/IP KeepAlive configuration value.
- DISABLE: Disable KeepAlive probing for this subsystem.
- 1 to 65534: Override the TCP/IP KeepAlive configuration value with the entered number of seconds. You should set this value close to the value in IDTHTOIN or the IRLM resource timeout value (IRLMRWT).

We recommend the default value.

► Pool thread timeout (POOLINAC)

This field specifies the approximate time, in seconds, that a DBAT can remain idle in the pool before it is terminated. A DBAT in the pool counts as an active thread against MAXDBAT and can hold locks, but does not have any cursors.

Specifying 0 causes a DBAT to terminate rather than go into the pool if the pool has a sufficient number of threads to process the number of inactive connections that currently exist. Specify 0 only if you are constrained on virtual storage below 2 GB in the DBM1 address space, as specifying 0 increases the likelihood of thread creation and corresponding overhead.

Two other fields of interest to DDF appears on DSNTIPN, the Tracing Parameters panel.

- DDF/RRSAF Accum (ACCUMACC)

The value you specify here determines whether DB2 will accumulate, or rollup, accounting records for DDF or RRSAF threads.

If you specify NO, DB2 writes an accounting record when a DDF thread is made inactive or when signon occurs for an RRSAF thread

If you specify a number, DB2 writes an accounting record in the specified interval for a given user, based on the aggregation fields specified in ACCUMUID.

- Aggregation fields (ACCUMUID)

The value you specify here determines which combination of user ID, transaction name, application name, and workstation name is used for the aggregation specified in ACCUMACC. This value also specifies whether strings of hex zeros, X'00', or blanks, X'40', are considered for rollup.

Refer to Table 7-4 on page 278 and related text for a discussion of accounting accumulation. Refer to *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846 for the details of setting ACCUMUID.

The following parameters are not in the install panels but do relate to DDF requests.

- PRGSTRIN

This parameter relates to progressive streaming for LOBs or XML and allows customers to disable progressive streaming behavior on DB2 for z/OS server where necessary. PK46079 added this parameter to the DSNZPARM macro DSN6FAC for DB2 9 for z/OS only. Refer to 5.7.8, “Progressive streaming” on page 222 for a discussion of this behavior.

- SQLINTRP

This parameter relates to the SQL interrupt function that was added to DB2 for z/OS V8. This parameter allows customers to disable the SQL interrupt function. PK59385 (DB2 9 for z/OS) and PK41661 (DB2 for z/OS V8) added this parameter to the DSNZPARM macro DSN6FAC. Refer to 5.7.9, “SQL Interrupts” on page 223 for a discussion of this behavior.

### 3.2.4 Updating the BSDS

Each of the parameters in the previous section that did not correspond to a DSNZPARM entry can be specified in the BSDS. These include LOCATION name, LUNAME, network PASSWORD, GENERIC LU name, DRDA port, secure port (SECPort) and resync port (RESPORT).

Beginning with DB2 9 for z/OS, you do not have to define DDF to VTAM, so LUNAME and GENERIC LU are not required. PASSWORD only applies to VTAM, and we do not recommend its use.

To define the BSDS when installing a DB2 subsystem, or to change the BSDS to reflect changes in your DDF configuration, you must run the change log inventory utility. We show the change log inventory job used for our standalone DB2 subsystem, DB9A, in Figure 3-22 on page 98. The RESPOR, PORT and SECPOR must all be different.

Remember, DSNJU003 can only be executed when the target DB2 subsystem is stopped.

```

//*****
//*          CHANGE LOG INVENTORY:
//*          UPDATE BSDS
//*****
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9A9.SDSNLOAD
//SYSUT1 DD DISP=OLD,DSN=DB9AU.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9AU.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF LOCATION=DB9A,LUNAME=SCPDB9A,
      RESPORT=12348,PORT=12347,SECPORT=12349
//*

```

Figure 3-22 DSNJU003 for DB9A BSDS

You can see your current settings for these DDF values from the Communication Record produced with the Print Log Map utility, DSNJU004. We show the output for DB9A in Figure 3-23. In DSNJU004 output, the resync port is RPORT and the secure port is SPORT.

```

**** DISTRIBUTED DATA FACILITY ****
          COMMUNICATION RECORD
          22:04:33 APRIL 22, 2009
LOCATION=DB9A IPNAME=(NULL) PORT=12347 SPORT=12349 RPORT=12348
ALIAS=(NULL)
IPV4=NULL IPV6=NULL
GRPIPV4=NULL GRPIPV6=NULL
LUNAME=SCPDB9A PASSWORD=(NULL) GENERICLU=(NULL)

```

Figure 3-23 DB9A DSNJU004 output with DDF values

### Updating the BSDS for a data sharing environment

In a data sharing environment the BSDS includes specifications for the data sharing group as well as the DDF specifications. When we began our project, the DSNJU003 job for the D9C1 member of our data sharing group looked like the example in Figure 3-24 on page 99. Note that we started with a minimal definition.

```

//*****
//*      CHANGE LOG INVENTORY:
//*      UPDATE BSDS
//*****
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF LOCATION=DB9C,LUNAME=SCPD9C1,
    RESPORT=38321,PORT=38320,SECPOR=0
DATASHR ENABLE
*
* WARNING! DO NOT CHANGE ANY PARAMETERS IN THE GROUP STATEMENT BELOW!
GROUP GROUPNAM=DB9CG,GROUPMEM=D9C1,MEMBERID=1
//*

```

Figure 3-24 DSNJU003 for D9C1 BSDS

The output from DSNJU004 in Figure 3-25 shows the Communication Record for the D9C1 member. There is no indication from the Communication Record that D9C1 is a member of a data sharing group other than the fact that the LOCATION name is one we know to be the LOCATION name of the group.

```

**** DISTRIBUTED DATA FACILITY ****
          COMMUNICATION RECORD
          21:55:59 APRIL 22, 2009
LOCATION=DB9C IPNAME=(NULL) PORT=38320 SPORT=NULL RPORT=38321
ALIAS=(NULL)
IPV4=NULL IPV6=NULL
GRPIPV4=NULL GRPIPV6=NULL
LUNAME=SCPD9C1 PASSWORD=(NULL) GENERICLU=(NULL)

```

Figure 3-25 D9C1 DSNJU004 output for member D9C1

After performing some of our test scenarios against the starting configuration, we enabled the Sysplex Distributor function of TCP/IP and defined dynamic virtual IP addresses (DVIPAs) for the members of the data sharing group. Refer to 3.1.4, “TCP/IP settings in a data sharing environment” on page 76 and subsequent sections to see the TCP/IP examples. We took advantage of the DB2 9 for z/OS function to specify the member DVIPA and group DVIPA in the BSDS.

We defined the group DVIPA, also known as the distributed DVIPA, as 9.12.4.102. We defined the member specific DVIPAs as follows

```

D9C1 9.12.4.103
D9C2 9.12.4.104
D9C3 9.12.4.105

```

Figure 3-26 shows the DSNJU003 input for D9C1 to change the DDF record to include the member DVIPA and the group DVIPA.

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
// DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C1.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF IPV4=9.12.4.103,GRPIPV4=9.12.4.102
```

Figure 3-26 DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C1

Figure 3-27 shows the corresponding input for member D9C2.

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
// DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C2.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C2.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF IPV4=9.12.4.104,GRPIPV4=9.12.4.102
//*
```

Figure 3-27 DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C2

Figure 3-28 shows the corresponding input for member D9C3.

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
// DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C3.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C3.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF IPV4=9.12.4.105,GRPIPV4=9.12.4.102
//*
```

Figure 3-28 DSNJU003 input to add DVIPA and Group DVIPA to the BSDS for D9C3



After running the DSNJU003 jobs, and restarting the corresponding DB2 members, the Communication Record from DSNJU004 shows the member DVIPA and the group DVIPA. See Figure 3-29 for the contents of the new Communication Record for member D9C1.

```

**** DISTRIBUTED DATA FACILITY ****
      COMMUNICATION RECORD
      21:39:14 APRIL 24, 2009
LOCATION=DB9C IPNAME=(NULL) PORT=38320 SPORT=NULL RPORT=38321
ALIAS=(NULL)
IPV4=9.12.4.103 IPV6=NULL
GRPIPv4=9.12.4.102 GRPIPv6=NULL
LUNAME=SCPD9C1 PASSWORD=(NULL) GENERICLU=(NULL)

```

Figure 3-29 DSNJU004 output for member D9C1 with DVIPA specified

We also issued the following command:

```
-D9C1 DISPLAY DDF
```

The output, in Figure 3-30, shows the group DVIPA and the member DVIPA. The SQL and RESYNC DOMAIN entries show -NONE.

```

RESPONSE=SC63
DSNL080I  -D9C1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DB9C              USIBMSC.SCPD9C1  -NONE
DSNL084I  TCPRT=38320  SECPRT=0      RESPT=38321  IPNAME=-NONE
DSNL085I  IPADDR=:9.12.4.102
DSNL086I  SQL      DOMAIN=-NONE
DSNL086I  RESYNC  DOMAIN=-NONE
DSNL089I  MEMBER IPADDR=:9.12.4.103
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Figure 3-30 Output from -D9C1 DISPLAY DDF showing DVIPA specifications

We added the domain name server (DNS) entries for the data sharing group, as described in 3.1.4, “TCP/IP settings in a data sharing environment” on page 76 and Figure 3-11 on page 81. Then we repeated the DDF display command. The output is shown in Figure 3-31.

```

DSNL519I  -D9C1 DSNLILNR TCP/IP SERVICES AVAILABLE 484
          FOR DOMAIN d9cg.itso.ibm.com AND PORT 38320
-D9C1 DISPLAY DDF
DSNL080I  -D9C1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 486
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DB9C              USIBMSC.SCPD9C1  -NONE
DSNL084I  TCPPORT=38320 SECPRT=0      RESPRT=38321 IPNAME=-NONE
DSNL085I  IPADDR=:9.12.4.102
DSNL086I  SQL      DOMAIN=d9cg.itso.ibm.com
DSNL086I  RESYNC  DOMAIN=d9cg.itso.ibm.com
DSNL089I  MEMBER IPADDR=:9.12.4.103
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Figure 3-31 Output from -D9C1 DISPLAY DDF showing DNS support for the group

This view of the output includes the message DSNL519I, which indicates DB2 is ready to accept connections on any IP address supported by the TCP/IP stack. This is the benefit of using DB2 9 for z/OS support to specify the group and member DVIPAs in the BSDS.

### Updating the BSDS for LOCATION ALIAS support

After we made the TCP/IP changes described in “Alias support” on page 82, we had to add the LOCATION ALIAS entries to our BSDS. Figure 3-32 shows the input for DSNJU003 to add two LOCATION ALIAS specifications to data sharing member D9C1. DB9CALIAS is a LOCATION ALIAS that only includes D9C1. DB9CSUBSET includes both D9C1 and D9C2.

```

//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
//        DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1  DD DISP=OLD,DSN=DB9CL.D9C1.BSDS01
//SYSUT2  DD DISP=OLD,DSN=DB9CL.D9C1.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
DDF      ALIAS=DB9CALIAS:38324,DB9CSUBSET:38325

```

Figure 3-32 DSNJU003 input to add ALIAS definitions to D9C1

Figure 3-33 shows the input for DSNJU003 to add LOCATION ALIAS, DB9CSUBSET, to data sharing member D9C2.

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB9C9.SDSNLOAD
// DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//SYSUT1 DD DISP=OLD,DSN=DB9CL.D9C2.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DB9CL.D9C2.BSDS02
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DDF ALIAS=DB9CSUBSET:38325
```

Figure 3-33 DSNJU003 input to add ALIAS definition to D9C2

After we restarted each of these two members, we ran DSNJU004 to verify the BSDS changes. Figure 3-34 shows the addition of two ALIAS specifications for D9C1.

```
**** DISTRIBUTED DATA FACILITY ****
      COMMUNICATION RECORD
      00:24:32 APRIL 28, 2009
LOCATION=DB9C IPNAME=(NULL) PORT=38320 SPORT=NULL RPORT=38321
ALIAS=DB9CALIAS:38324,DB9CSUBSET:38325
IPV4=9.12.4.103 IPV6=NULL
GRPIPV4=9.12.4.102 GRPIPV6=NULL
LUNAME=SCPD9C1 PASSWORD=(NULL) GENERICLU=(NULL)
```

Figure 3-34 DSNJU004 output showing LOCATION ALIAS and DVIPA with IPV4

### 3.2.5 DDF address space setup

The DB2 distributed services address space (ssidDIST) startup JCL is generated by the installation job DSNTIJMV (if the DDF startup option is set to AUTO or COMMAND, and a location name is specified on the DSNTIPR panel). The DSNTIJMV job can be found in the hlq.SDSNSAMP data set. Figure 3-35 on page 104 shows the JCL for D9C1DIST, one of the members of our data sharing group.

```

//D9C1DIST JOB MSGLEVEL=1
//STARTING EXEC D9C1DIST
XX*****
XX*      JCL PROCEDURE FOR THE STARTUP OF THE
XX*      DISTRIBUTED DATA FACILITY ADDRESS SPACE
XX*
XX*****
XXD9C1DIST PROC RGN=OK,
XX          LIB='DB9C9.SDSNEXIT'
XXIEFPROC EXEC PGM=DSNYASCP,REGION=&RGN
IEFC653I SUBSTITUTION JCL - PGM=DSNYASCP,REGION=OK
XXSTEPLIB DD DISP=SHR,DSN=&LIB
IEFC653I SUBSTITUTION JCL - DISP=SHR,DSN=DB9C9.SDSNEXIT
XX      DD DISP=SHR,DSN=CEE.SCEERUN
XX      DD DISP=SHR,DSN=DB9C9.SDSNLOAD

```

Figure 3-35 JCL for D9C1DIST

### 3.2.6 Stored procedures and support for JDBC and SQLJ

If you need to provide support for JDBC and SQLJ requesters for the first time, there are several steps you must complete. Most of these steps are for DB2 for z/OS as the DRDA AS, but some relate to the requesters. The steps to install support for JDBC and SQLJ are as follows:

1. Allocate and load IBM Data Server Driver for JDBC and SQLJ libraries. Perform this step on the clients.
2. On DB2 for z/OS, set the DESCSTAT parameter to YES (DESCRIBE FOR STATIC on the DSNTIPF installation panel). This is necessary for SQLJ support.
3. In z/OS UNIX System Services, edit the .profile file to customize environment variable settings.
4. (Optional) Customize IBM Data Server Driver for JDBC and SQLJ configuration properties. This refers to the properties on the clients.
5. On DB2 for z/OS, enable the DB2-supplied stored procedures and define the tables that are used by the IBM Data Server Driver for JDBC and SQLJ.
6. In z/OS UNIX System Services, run the DB2Binder utility to bind the packages for the IBM Data Server Driver for JDBC and SQLJ.
7. Install z/OS Application Connectivity to DB2 for z/OS feature. This applies if you have a JDBC or SQLJ application on z/OS in an LPAR where you do not have a DB2 for z/OS subsystem. This feature is effectively the Type 4 driver for z/OS and provides the DRDA AR function without a local DB2 (and DDF) to communicate with a DB2 for z/OS server elsewhere in the network.

Refer to *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846 for more information about these steps.

## 3.3 Workload Manager setup

Workload Manager (WLM) is the operating system component responsible for managing all the work in your z/OS environment. WLM uses workload classification, service classes, and performance objectives to manage the resources of the z/OS system to meet your business objectives. From a DB2 perspective, WLM manages the priority of the DB2 address spaces, including DDF, IRLM, and stored procedure address spaces.

WLM plays an important role in managing DDF work. We refer to work that enters the system through DDF as DDF transactions. WLM manages the priority of DDF transactions separately from the priority of the DDF address space. Define your DDF transactions to WLM correctly to ensure that the DDF transactions receive an appropriate level of system resources and can achieve the performance objectives that meet your business requirements.

In this section we discuss how WLM uses enclaves to manage your DDF transactions, and we show how to define performance objectives for your DDF transaction workload.

### 3.3.1 Enclaves

An enclave is an independently dispatchable unit of work, or a business transaction, that can span multiple address spaces and can include multiple SRBs and TCBs. DDF transactions execute as enclaves in DB2. (DB2 also uses enclaves for other purposes, such as parallel queries or native SQL procedures.)

The DDF address space owns all the enclaves created by the distributed data facility. However, there is no special connection between the enclaves and their owner address space. Each enclave is managed separately by the MVS System Resource Manager (SRM) according to its performance objectives.

#### DDF and the life of an enclave

When a connection request comes to DDF, the connection must be associated with a DBAT before the DDF transaction can execute SQL. When the DDF transaction processes its first SQL statement, DDF calls WLM to create an enclave. WLM then manages the enclave based on the workload characteristics assigned to that enclave.

The enclave is the basis for assigning system resources to the DDF transaction running on that enclave. The enclave is also the basis for reporting thread performance. Refer to 7.2.1, “Database Access Threads” on page 271 for details on DDF thread performance.

When the enclave is deleted depends on whether the DBAT can become pooled. If the DBAT becomes pooled, the enclave is deleted. If the DBAT cannot become pooled, the enclave is only deleted at thread termination time. (If a DBAT becomes type 1 inactive, the enclave is deleted. Type 1 inactive DBATs generally only apply to DB2 Private Protocol connections.)

### 3.3.2 Managing DDF work with WLM

DDF transactions are classified and defined to a WLM service class in the active WLM policy. Use the WLM administrative panels to define service classes, report classes, and classification rules for DDF transactions.

To define DDF transactions to WLM, perform the following tasks:

- ▶ Define service classes, performance objectives for these service classes, and optional report classes.
- ▶ Define classification rules to assign incoming DDF transactions to the appropriate service class. DDF transactions are classified using the subsystem type of DDF, or SUBSYS = DDF.
- ▶ Activate the WLM policy.

**Attention:** If you do not define classification rules for SUBSYS = DDF, all your DDF transactions will run in the SYSOTHER service class, which has a discretionary goal. This means your DDF transactions will only execute after all the other service calls goals are met. In a busy system, this could mean that your DDF transactions get little service.

The DIST address space is not managed based on the same classification rules for DDF transactions. DIST performance objectives are defined with SUBSYS = STC. Generally DIST should be defined to run with the same objectives as MSTR and DBM1, all of which should be higher than the performance objectives of the DDF enclaves.

### ***Performance objectives for DDF transactions***

For a DDF transaction, WLM assigns the performance goal to the enclave, and it is the lifetime of the enclave that WLM takes as the duration of the work. Therefore, when you run with CMTSTAT=INACTIVE, DDF creates one enclave per transaction, and response time goals and multiple period objectives can be used. However, if you have CMTSTAT=ACTIVE DDF creates one enclave for the life of the thread, and response time goals and multiple periods should not be used.

If you specify CMSTAT=INACTIVE but the DBAT cannot be pooled at commit, then the enclave is not deleted and may end up running against lower period objectives. You should ensure this happens only infrequently. See also 7.2.1, “Database Access Threads” on page 271.

### **Modifying WLM definitions for DB2 and DDF**

The remainder of this section describes the changes we made to the WLM definitions for our DB2 address spaces and the specifications for DDF transactions in our environment.

For details on WLM, refer to *System Programmer's Guide to: Workload Manager*, SG24-6472.

We started by modifying the classification rules for DB2. By default, DB2 address spaces run in SYSSTC. Like many customers, we prefer running our DB2 address spaces in importance 1 with a reasonable velocity goal. We choose option 6. Classification rules in Figure 3-36 on page 107 to modify the rules for subsystem type STC.

```

-----
Functionality LEVEL019          Definition Menu          WLM App1 LEVEL021
Command ==> _____

Definition data set . . . : none

Definition name . . . . . soaredb   (Required)
Description . . . . . _____

Select one of the
following options. . . . . 6__  1.  Policies
                                2.  Workloads
                                3.  Resource Groups
                                4.  Service Classes
                                5.  Classification Groups
                                6.  Classification Rules
                                7.  Report Classes
                                8.  Service Coefficients/Options
                                9.  Application Environments
                                10. Scheduling Environments

```

Figure 3-36 WLM: Choosing Classification Rules

Refer to Figure 3-37, where you can see that the default service class for subsystem type STC, Started Tasks, is STC and that a report class of RSYSDFLT is defined.

Subsystem Type Selection List for Rules			Row 1 to 14 of 14	
Command ==>				
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, /=Menu Bar				
			-----Class-----	
Action	Type	Description	Service	Report
—	ASCH	APPC Transaction Programs	ASCH	
—	CB	WebSphere/Component Broker	DDFDEF	RCB
—	CICS	CICS Transactions	CICS	
—	DB2	DB2 Sysplex Queries	DB2QUERY	
—	DDF	DDF Work Requests	DDFBAT	
—	EWLM	EWLM Subsystem for ESC/ETC	SC_EWLM	REWLMDEF
—	IMS	IMS Transactions	IMS	
—	IWEB	Web Work Requests		RIWEB
—	JES	Batch Jobs	BATCHLOW	BATCHDEF
—	MQ	MQSeries Workflow		RMQ
—	OMVS	UNIX System Services	SYSTC1	
3_	<b>STC</b>	<b>Started Tasks</b>	<b>STC</b>	<b>RSYSDFLT</b>
—	SYSH	linux		
—	TSO	TSO Commands	TSO	TSO

Figure 3-37 WLM: Subsystem Type Selection: Choosing classification rules for started tasks

Figure 3-38 shows the changes we made to subsystem type STC. We added a transaction name group (TNG) for the DB2 address spaces, which we called DB2AS, with service class STCHI and report class RDB2AS. We placed the definition for this TNG before the SPM definitions, otherwise the SPM would override our DB2AS TNG specifications. We also modified the existing transaction (TN) definition for the IRLM address spaces. We made sure the IRLMs had a service class of SYSSTC, because they should be equal to or higher than the DB2 address spaces.

-----

Modify Rules for the Subsystem Type

Row 1 to 8 of 17

Command ==>

Scroll ==> CSR

Subsystem Type . : STC

Fold qualifier names? Y (Y or N)

Description . . . Started Tasks

Action codes:

A=After

C=Copy

M=Move

I=Insert rule

B=Before

D=Delete row

R=Repeat

IS=Insert Sub-rule

More ==>

-----Qualifier-----

-----Class-----

Action

Type

Name

Start

Service

Report

DEFAULTS:

STC

RSYSDFLT

1

TNG

DB2AS

STCHI

RDB2AS

1

TN

%%%IRLM

SYSSTC

RIRLM

1

SPM

SYSTEM

SYSTEM

RSYSTEM

1

SPM

SYSSTC

SYSSTC

RSYSSTC

1

TN

DFS

SYSSTC2

RINETD1

1

TN

DFSKERN

SYSSTC3

RINETD1

1

TN

INETD1

SYSSTC4

RINETD1

1

TN

PMPAP

SYSSTC5

RMPAP

Figure 3-38 WLM: STC service classes and report classes

Figure 3-39 shows what we included in our DB2AS TNG. All of the DB2 subsystem or member address spaces run in service class STCHI.

\* Transaction Name Group DB2AS - DB2 system address spaces

Created by user PAOLOR6 on 2009/04/15 at 13:17:45

Last updated by user PAOLOR6 on 2009/04/15 at 14:28:09

Qualifier

name

Description

-----

%%%MSTR

System services

%%%DBM1

Database services

%%%DIST

DDF

Figure 3-39 WLM: Transaction Name Group (TNG) for all DB2s in our sysplex



Figure 3-40 shows the performance objective for service class STCHI.

* Service Class STCHI - Started Tasks			
Created by user FISCHER on 2006/12/11 at 18:20:50			
Base last updated by user PAOLOR6 on 2009/04/15 at 13:03:42			
Base goal:			
CPU Critical flag: NO			
#	Duration	Imp	Goal description
-	-	-	-
1		1	Execution velocity of 60

Figure 3-40 WLM: STCHI service class goal

STCHI runs with importance 1 and velocity goal of 60. This is an appropriate goal for DB2 address spaces. All the DDF transactions we define below run below importance 1. It is not a good idea for transactions to be running equal with the subsystem address spaces. The IRLM is running in SYSSTC, which is above importance 1.

Figure 3-41 shows a portion of the System Display and Search Facility (SDSF) Display Active (DA) panel for prefix D9C1. You can see that the IRLM is in service class SYSSTC and the other DB2 address spaces are in service class STCHI.

-----									
SDSF	DA	SC63	SC63	PAG	0	CPU/L/Z	7/	7/	0
LINE 1-4 (4)									
COMMAND INPUT ==>									SCROLL ==> CSR
NP	JOBNAME	Workload	SrvClass	SP	ResGroup	Server	Quiesce	ECPU-Time	ECPU%
	D9C1DBM1	STCTASKS	<b>STCHI</b>	1		NO		21.35	0.00
	D9C1DIST	STCTASKS	<b>STCHI</b>	1		NO		2.90	0.00
	D9C1IRLM	SYSTEM	<b>SYSSTC</b>	1		NO		352.58	0.00
	D9C1MSTR	STCTASKS	<b>STCHI</b>	1		NO		176.93	0.00

Figure 3-41 SDSF display showing service classes for D9C1 address spaces

## Defining service classes for DDF transactions

Choose option 4 from the main WLM menu to define Service Classes. In our environment we started with three service classes for DDF transactions, DDFONL for high priority transactions, DDFDEF, which at one time had been the default, and DDFBAT, which was the default service class when we started. Figure 3-42 on page 110 shows the service class definition for DDFONL.

```

* Service Class DDFONL - DDF High priority

Created by user BARTR2 on 2003/02/18 at 17:09:58
Base last updated by user BART on 2003/04/11 at 17:42:53

Base goal:
CPU Critical flag: NO

#   Duration   Imp   Goal description
-   -
1   500        2    90% complete within 00:00:01.000
2           3    Execution velocity of 40

```

Figure 3-42 WLM: DDFONL service class goals

Note that DDF transactions that run in DDFONL service class can consume 500 service units in first period, during which time they run at a WLM importance of 2. The goal is that 90% of DDF transactions in this service class will complete within one second. Any DDF transaction that consumes more than 500 service units will run at a WLM importance of 3 with a velocity goal of 40.

Figure 3-43 shows the DDFDEF service class, which has lower performance objectives. First period transactions run at importance 3 and we expect 80% to complete within half a second. After 500 service units they run in second period at importance 4 with a velocity goal of 20.

```

Browse                               Line 00000000 Col 001 072
Command ==>                          SCROLL ==> CSR
***** Top of Data *****
* Service Class DDFDEF - DDF Requests

Created by user BART on 2003/04/11 at 20:27:56
Base last updated by user BART on 2003/04/11 at 20:27:56

Base goal:
CPU Critical flag: NO

#   Duration   Imp   Goal description
-   -
1   500        3    80% complete within 00:00:00.500
2           4    Execution velocity of 20

***** Bottom of Data *****

```

Figure 3-43 WLM: DDFDEF service class goals

Figure 3-44 shows the DDFBAT service class which is the default service class for workload of subsystem type DDF in our environment.

* Service Class DDFBAT - DDF low priority			
Class assigned to resource group DDF			
Created by user BARTR2 on 2003/02/18 at 17:12:04			
Base last updated by user PAOLOR6 on 2009/04/15 at 20:12:47			
Base goal:			
CPU Critical flag: NO			
#	Duration	Imp	Goal description
-	-	-	-
1	1000	3	Average response time of 00:30:00.000
2		4	70% complete within 01:00:00.000

Figure 3-44 WLM: DDFBAT service class goals for our default DDF service class

Remember to define a default service class for subsystem type DDF in case DDF transactions that you do not otherwise define come into the system. You would not want these to end up in service class SYSOTHER.

We added two service classes for specific scenarios. These are DDFTOT for transactions beginning with characters “ToTo” and DDFTST.

Figure 3-45 shows the service class definition for DDFTOT.

* Service Class DDFTOT - DDF TOT0 for RB			
Created by user PAOLOR6 on 2009/04/22 at 20:05:12			
Base last updated by user PAOLOR6 on 2009/04/22 at 20:05:12			
Base goal:			
CPU Critical flag: NO			
#	Duration	Imp	Goal description
-	-	-	-
1	500	2	80% complete within 00:00:00.500
2		3	Execution velocity of 20

Figure 3-45 WLM: DDFTOT service class definition

Figure 3-46 shows the service class definition for DDFTST. We had the same performance objectives but defined a separate service class to support separate scenarios.

* Service Class DDFTST - DDF Test for RB			
Created by user PAOLOR6 on 2009/04/22 at 20:01:31			
Base last updated by user PAOLOR6 on 2009/04/22 at 20:03:05			
Base goal:			
CPU Critical flag: NO			
#	Duration	Imp	Goal description
-	-----	-	-----
1	500	2	80% complete within 00:00:00.500
2		3	Execution velocity of 20

Figure 3-46 WLM: DDFTST service class definition

## Defining classification rules

Once you have defined the service classes, and optional report classes, that you require, you can set up the classification rules that assign the incoming DDF transactions to the appropriate service classes.

There are many attributes or qualifiers you can use to classify DDF transactions and assign them to a service class. Table 3-5 provides a list of the attributes or qualifiers available to classify DDF transactions. The attributes with an asterisk, (\*), identify those fields that can be set by client information APIs, for example SQLSET commands or Java SET\_CONNECTION\_ATTRIBUTES.

**Important:** To classify work (transactions) coming in through DDF, use the DDF subsystem type. The DB2 subsystem type is only for DB2 sysplex query parallelism. In addition, to assign performance goals for the DB2 address spaces (including ssidDIST), you use the STC subsystem type, as we describe in Figure 3-38 on page 108.

Table 3-5 DDF work classification attributes

Attribute	Type	Description
Accounting Information*	AI	Can be passed from an application through Client Information APIs
Correlation Information*	CI	Application program by default, but application can set through Client Information APIs
Collection Name	CN	Collection name of the first SQL package accessed by the requester in the unit of work
Connection Type	CT	Always 'DIST' for DDF server threads
Package Name	PK	Name of the first DB2 package accessed by the DRDA requester in the unit of work
Plan Name*	PN	Always 'DISTSERV' for DDF server threads accessed through DRDA requesters
Procedure Name	PR	Name of the procedure called as the first request of the unit of work

Attribute	Type	Description
Process Name*	PC	Client application name by default, but can be set through Client Information APIs
Subsystem Collection Name	SSC	Usually the DB2 data sharing group name
Subsystem Instance	SI	DB2 server's MVS subsystem name
Sysplex Name	PX	Name assigned to the parallel sysplex at IPL
Userid	UI	DDF server thread's primary AUTHID
Subsystem Parameter *	SPM	Concatenation of client user ID and workstation

\* The attribute can be set by Client Information APIs.

Figure 3-47 shows a subset of the classification rules that apply to our standalone subsystem, DB9A, and to the members of our data sharing group.

* Subsystem Type DDF - DDF Work Requests					
Last updated by user YIM on 2009/05/21 at 15:23:39					
Classification:					
Default service class is DDFBAT					
There is no default report class.					
	Qualifier # type	Qualifier name	Starting position	Service Class	Report Class
----	-----	-----	-----	-----	-----
----	1 SI	DB9A		DDFDEF	RDB9ADEF
	2 . PC	. TRX*		DDFONL	RSSL
	2 . AI	. TotoA*	56	DDFTOT	
	2 . UI	. PAOLOR3			RNISANTI
	1 SI	D9C*		DDFDEF	RD9CG
	2 . UI	. PAOLOR3			RNMSHARE
	2 . UI	. PAOLOR7		DDFTST	
	2 . PC	. db2j*		DDFONL	

Figure 3-47 A subset of WLM classification rules

We had two levels of classification rules for DB9A and two levels for the members of our data sharing group.

Any DDF transactions that requested connections from DB9ADIST were qualified based on the first four lines.

- ▶ DDF transactions with an application name (PC) beginning with the characters "TRX" ran in service class DDFONL and we could report on them separately with report class RSSL.
- ▶ Requesters that passed accounting information (AI) with the characters "ToToA" beginning in position 56 of the accounting string ran in service class DDFTOT. We did not need a separate report class in this case, because we knew only those transactions would be in DDFTOT service class.
- ▶ Any DDF transactions with a primary AUTHID (UI) of PAOLOR3 ran in the default service class for DDF (DDFBAT, not shown on this panel) with a separate report class.

- Any other DDF transactions that came to DB9ADIST (SI = DB9A) ran in service class DDFDEF with report class RDB9ADEF

Any DDF transactions that requested connections to any of the data sharing group members were qualified based on the last four lines,

- Any DDF transactions with a primary AUTHID (UI) of PAOLOR3 ran in the default service class for DDF (DDFBAT, not shown on this panel) with a separate report class.
- Any DDF transactions with a primary AUTHID (UI) of PAOLOR7 ran in the DDFTST service class.
- DDF transactions with an application name (PC) beginning with the characters “db2j” ran in service class DDFONL.
- Any other DDF transactions that came to a data sharing group member (SI beginning with ‘D9C’) ran in service class DDFDEF with report class RD9CG.

This set of classification rules gave us the ability to distinguish between specific workloads and monitor DDF transaction either with service class, for example from the SDSF DA panel, or with report classes.

**Note:** The classification rules are evaluated in the order they are displayed by the WLM ISPF interface. For good performance, especially if you set up many classification rules, specify the ones that are either the most important or the most likely to be used, at the beginning of the list.

**Tip:** It is worth noting that you can start the matching for a certain qualifier in any position of the string you are matching against. You indicate the starting position in the “Start position” column. The default is to start matching from the beginning of the string. If your organization has good naming standards in place, this can be a powerful way to distinguish between different types of work.

It is also worth noting that the values are case sensitive. Be careful when editing the WLM panels that attributes you intend to be lower case are not folded to upper case.

## 3.4 DB2 for LUW to DB2 for z/OS setup

In this section we briefly describe the steps to connect a DB2 for LUW DRDA requester to DB2 for z/OS. We describe how to configure the IBM Data Server Clients or Drivers to connect directly to DB2 for z/OS. We then describe briefly DB2 Connect configuration commands in case you still support DB2 Connect in your environment.

### 3.4.1 IBM Data Server Drivers and Clients

If you use the IBM Data Server Client, we recommend you use the Client Configuration Assistant (CCA) if possible, as this reduces the likelihood of errors. The CCA is a GUI tool that guides you through the configuration process.

If you use the IBM Data Server Runtime Client or one of the non-Java IBM Data Server Drivers, you must create and populate the `db2dsdriver.cfg` file.

If you are migrating from a DB2 Connect environment you can issue the **db2dsgc fgi11** command, which will create the `db2dsdriver.cfg` file with most of the information you need. This file will be in the `cfg` directory in your DB2 instance home. If you if you do not have a `cfg`

directory under your instance home, the db2dsdcfgfill command will fail. Create the directory before executing the command. Refer to 6.2.2, “Application Servers” on page 249 for a description of the use of this command in a data sharing environment.

If you are not migrating from a DB2 Connect environment, you can use the sample db2dsdriver.cfg provided with the driver. We include an example from our environment in Figure 3-48.

```
<configuration>
  <DSN_Collection>
    <dsn alias="DB9C_DIR" name="DB9C" host="wtsc63.itso.ibm.com"
port="38320"/>
    <!-- Long aliases are supported -->
  </dsn>
</DSN_Collection>
<databases>
  <database name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
    <WLB>
      <parameter name="enableWLB" value="true"/>
      <parameter name="maxTransports" value="100"/>
    </WLB>
    <ACR>
      <parameter name="enableACR" value="true"/>
    </ACR>
  </database>
</databases>
<parameters>
  <parameter name="enableDirectXA" value="true"/>
</parameters>
</configuration>
```

Figure 3-48 Sample db2dsdriver.cfg for our environment

In Figure 3-49 on page 116 we show the sample db2dsdriver.cfg file provided with the drivers. Key information you will need to customize the <DSN Collection: for your environment is as follows:

- ▶ name = LOCATION name of the DB2 subsystem or data sharing group
- ▶ host = IP address (group DVIPA in data sharing) or domain name
- ▶ port = SQL port

```

<configuration>
  <DSN_Collection>
    <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001"/>
    <!-- Long aliases are supported -->
    <dsn alias="longaliasname2" name="name2" host="server2.net1.com"
port="55551">
      <parameter name="Authentication" value="Client"/>
    </dsn>
  </DSN_Collection>
  <databases>
    <database name="name1" host="server1.net1.com" port="50001">
      <parameter name="CurrentSchema" value="OWNER1"/>
      <WLB>
        <parameter name="enableWLB" value="true"/>
        <parameter name="maxTransports" value="50"/>
      </WLB>
      <ACR>
        <parameter name="enableACR" value="true"/>
      </ACR>
    </database>
    <!-- Local IPC connection -->
    <database name="name3" host="localhost" port="0">
      <parameter name="IPCInstance" value="DB2"/>
      <parameter name="CommProtocol" value="IPC"/>
    </database>
  </databases>
  <parameters>
    <parameter name="GlobalParam" value="Value"/>
  </parameters>
</configuration>

```

Figure 3-49 Sample db2dsdriver.cfg provided with the driver

For the IBM Data Server Driver for JDBC and SQLJ, you can specify the DB2 for z/OS server using either the DriverManager or DataSource interface.

Refer to Chapter 5, “Application programming” on page 185 for more information about defining the IBM Data Server Drivers to connect to DB2 for z/OS.

### 3.4.2 DB2 Connect

If you are supporting DB2 Connect and have not yet migrated to the IBM Data Server Drivers (or Clients) and you have the DB2 Connect modules installed on the workstation, you can enter the commands described below with the Command Line Processor (CLP), or you can use the Client Configuration Assistant (CCA).

#### Two-tier connection to DB2 for z/OS

In a two-tier configuration, the application runs on a workstation and communicates directly to DB2 for z/OS. Other than network routers, no hardware is between the workstation and DB2 for z/OS. In the following discussion we assume a workstation is connecting to LOCATION DB9A, our standalone DB9A subsystem.



Figure 3-50 shows a diagram of a two-tier configuration in our environment with the commands required to define the connection.

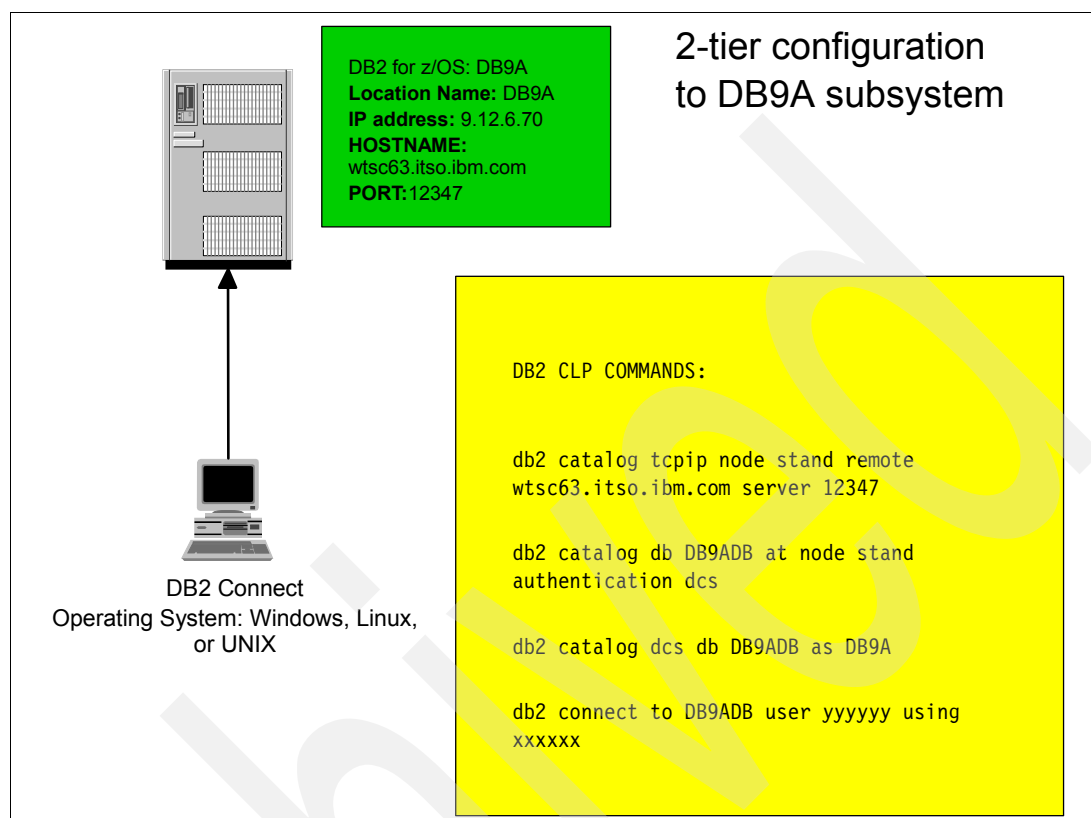


Figure 3-50 Two-tier configuration to our standalone DB2 for z/OS, DB9A

Table 3-6 shows the commands on the left, and the information about the DB2 for z/OS environment required on the right.

Table 3-6 Commands to connect 2-tier to DB2 for z/OS and the information required

Commands to issue for DB2 Connect	Information from DB2 for z/OS environment
<b>db2 catalog tcpip node stand remote wtsc63.itso.ibm.com server 12347</b> <ul style="list-style-type: none"> <li>► 'stand' is an arbitrary name for this node (standalone DB2 subsystem). It links the tcpip node to the database in the next command</li> <li>► The IP address could be used instead of host name</li> <li>► Specify the DRDA port number, or service name, after the 'server' keyword</li> </ul>	<ul style="list-style-type: none"> <li>► z/OS IP host name (or IP address) = wtsc63.itso.ibm.com</li> <li>► DRDA port number for DB2 for z/OS = 12347</li> </ul>
<b>db2 catalog db DB9ADB at node stand authentication dcs</b> <ul style="list-style-type: none"> <li>► Other security options are available besides 'authentication dcs'</li> </ul>	<ul style="list-style-type: none"> <li>► DB9ADB is an arbitrary name you will use to connect to DB2 for z/OS.</li> </ul>
<b>db2 catalog dcs db DB9ADB as DB9A</b>	<ul style="list-style-type: none"> <li>► LOCATION name for DB2 for z/OS = DB9A</li> </ul>
<b>db2 connect to DB9ADB user yyyyyy using xxxxxx</b>	<ul style="list-style-type: none"> <li>► user ID = yyyyyy ; password = xxxxxx</li> </ul>

For a description of other security options, refer to Chapter 4, “Security” on page 129.

### Three-tier connection to DB2 for z/OS

In a three-tier configuration, the application runs on a workstation that communicates with a server or gateway that runs DB2 Connect server. The DB2 Connect server communicates to DB2 for z/OS. The workstation can have one of the IBM Data Server Drivers or Clients installed. In the following discussion we assume a workstation is connecting to LOCATION DB9A, our standalone DB9A subsystem.

Figure 3-51 shows a diagram of a three-tier configuration in our environment with the commands required to define the connection from the DB2 Connect server to DB2 for z/OS and from the workstation to the DB2 Connect server.

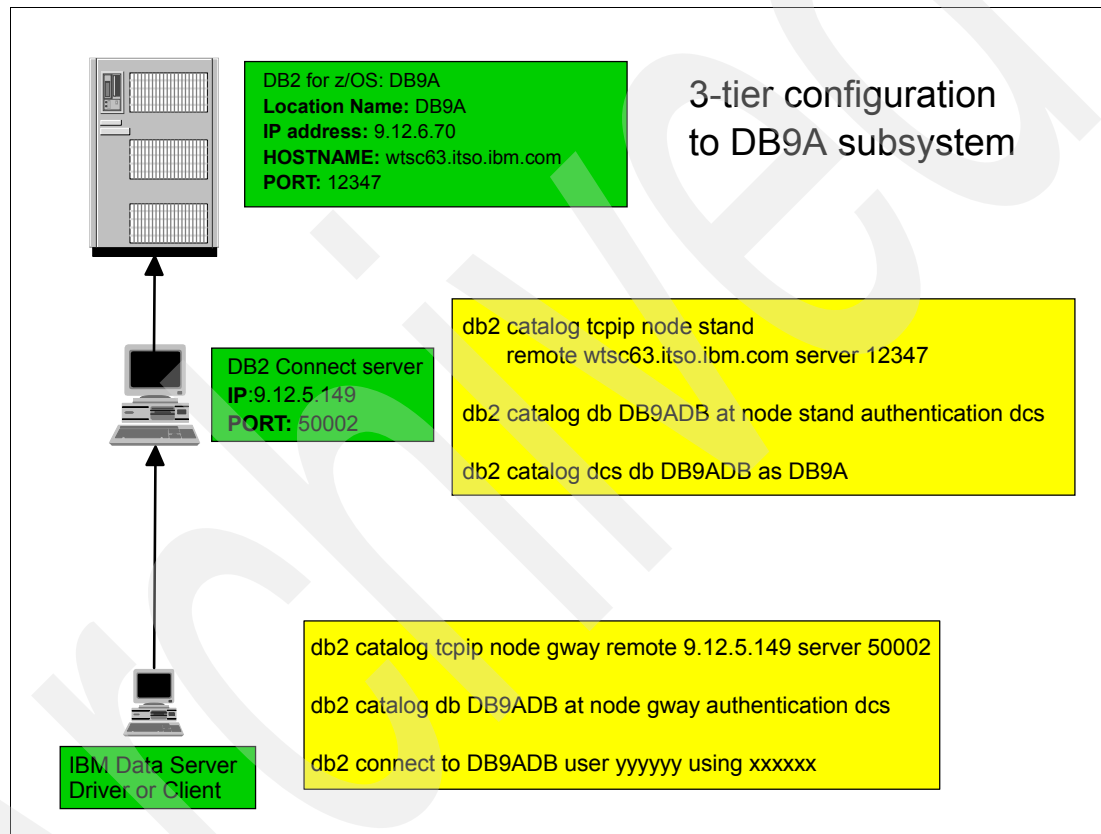


Figure 3-51 Three-tier configuration to our standalone DB2 for z/OS

Table 3-7 on page 119 shows the commands on the left, and the information about the DB2 Connect server environment required on the right. The information about DB2 for z/OS does not change, so the commands entered at the DB2 Connect server for the three-tier configuration are the same as those entered in the two-tier configuration in Table 3-6 on page 117.

Table 3-7 Commands to connect 3-tier to DB2 for z/OS and the information required

Commands to issue for workstation client	Information from DB2 Connect server environment (DB2 for z/OS does not change)
<b>db2 catalog tcpip node gway remote 9.12.5.149 server 50002</b> Notes: <ul style="list-style-type: none"> <li>► 'gway' is an arbitrary name for DB2 Connect server node (gateway).</li> <li>► We show the IP address. The host name could be used instead.</li> <li>► Specify the DRDA port number, or service name, after the 'server' keyword</li> </ul>	<ul style="list-style-type: none"> <li>► z/OS IP address) = 9.12.5.149</li> <li>► DRDA port number for DB2 Connect server = 50002</li> </ul>
<b>db2 catalog db DB9ADB at node gway authentication dcs</b> Notes: <ul style="list-style-type: none"> <li>► Other security options are available besides 'authentication dcs'</li> </ul>	<ul style="list-style-type: none"> <li>► DB9ADB is an arbitrary name you will use to connect to DB2 for z/OS.</li> </ul>
<b>db2 catalog dcs db DB9ADB as DB9A</b>	<ul style="list-style-type: none"> <li>► LOCATION name for DB2 for z/OS = DB9A</li> </ul>
db2 connect to DB9ADB user yyyyyy using xxxxxx	<ul style="list-style-type: none"> <li>► user ID = yyyyyy ; password = xxxxxx</li> </ul>

### Connection to a data sharing group

For the most part, connecting to a DB2 data sharing group requires the same commands. The information will be different. Best practices include specifying the group DVIPA or group domain name and allowing the Sysplex Distributor to route the connection request to an available member of the data sharing group. The diagrams and tables assume that the client application is connecting to a member of DB2 data sharing group D9CG, which has a LOCATION of DB9C and members D9C1, D9C2 and D9C3.

Figure 3-52 on page 120 shows the two-tier configuration with commands to connect to a member of our data sharing group.

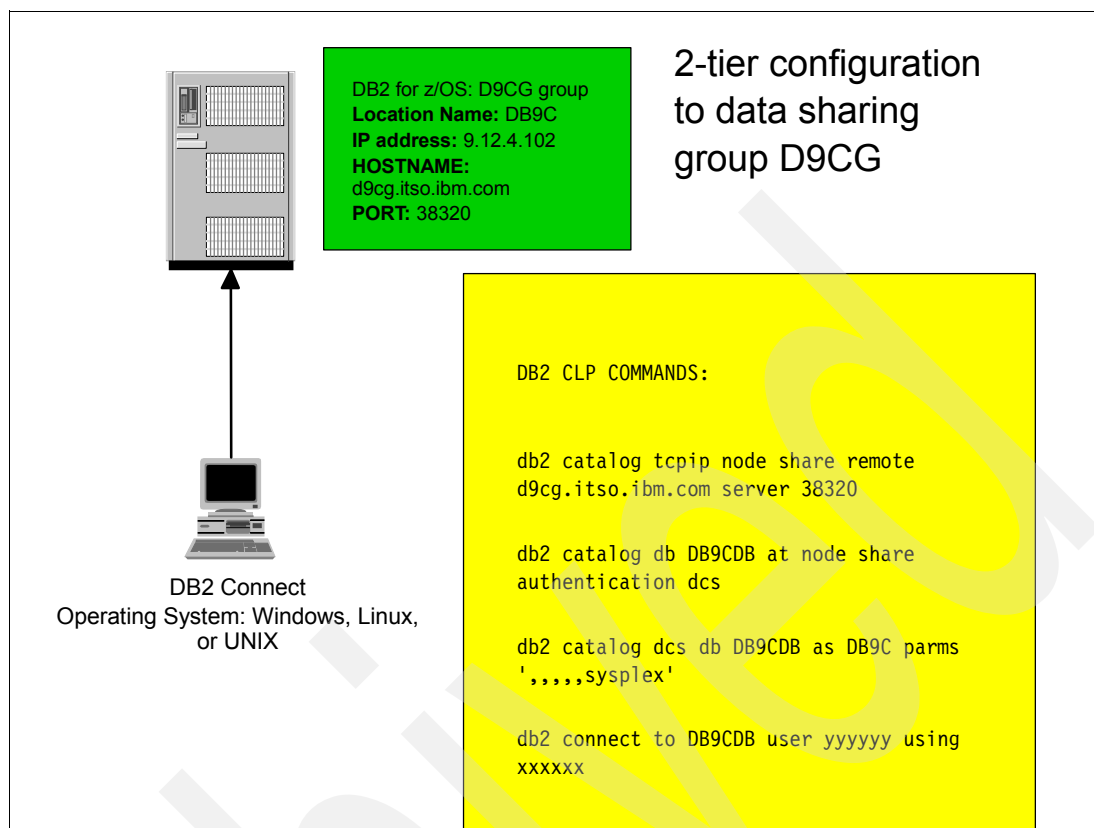


Figure 3-52 Two-tier connection to our DB2 for z/OS data sharing group

Table 3-8 shows the commands on the left, and the required information to connect to the DB2 for z/OS data sharing environment on the right.

Table 3-8 Commands to connect 2-tier to DB2 data sharing group and the information required

Commands to issue for DB2 Connect	Information from DB2 for z/OS environment
<b>db2 catalog tcpip node share remote d9cg.itso.ibm.com server 38320</b> Notes: <ul style="list-style-type: none"> <li>► 'share' is an arbitrary name for this node (data sharing group). It links the tcpip node to the database in the next command</li> <li>► The IP address (group DVIPA) could be used instead of group host name</li> <li>► Specify the DRDA port number, or service name, after the 'server' keyword</li> </ul>	<ul style="list-style-type: none"> <li>► z/OS IP host name (or IP address) = d9cg.itso.ibm.com. This is the domain name for the data sharing group.</li> <li>► DRDA port number for the DB2 for z/OS data sharing group = 38320</li> </ul>
<b>db2 catalog db DB9CDB at node share authentication dcs</b> Note: <ul style="list-style-type: none"> <li>► Other security options are available besides 'authentication dcs'</li> </ul>	<ul style="list-style-type: none"> <li>► DB9CDB is an arbitrary name to connect to the DB2 for z/OS data sharing group.</li> </ul>
<b>db2 catalog dcs db DB9CDB as DB9C parms ',,,,,sysplex'</b> Note: <ul style="list-style-type: none"> <li>► parms ',,,,,sysplex' specifies sysplex support, including workload balancing</li> </ul>	<ul style="list-style-type: none"> <li>► LOCATION name for DB2 for z/OS data sharing group = DB9C</li> </ul>

Commands to issue for DB2 Connect	Information from DB2 for z/OS environment
db2 connect to DB9CDB user yyyyyy using xxxxxx	► user ID = yyyyyy ; password = xxxxxx

If you use LOCATION ALIAS support for your data sharing group, you can choose to catalog some of your clients to the group DRDA port number and LOCATION name and others to the subset or alias port number and LOCATION ALIAS. In the above diagram and table, that would mean one of the following changes:

- For the single-member ALIAS, change the LOCATION from DB9C to DB9CALIAS and the port from 38320 to 38324.
- For the two-member ALIAS, change the LOCATION from DB9C to DB9CSUBSET and the port from 38320 to 38325.

Figure 3-53 shows a diagram of a three-tier configuration in our environment with the commands required to define the connection from the DB2 Connect server to our DB2 for z/OS data sharing group and from the workstation to the DB2 Connect server.

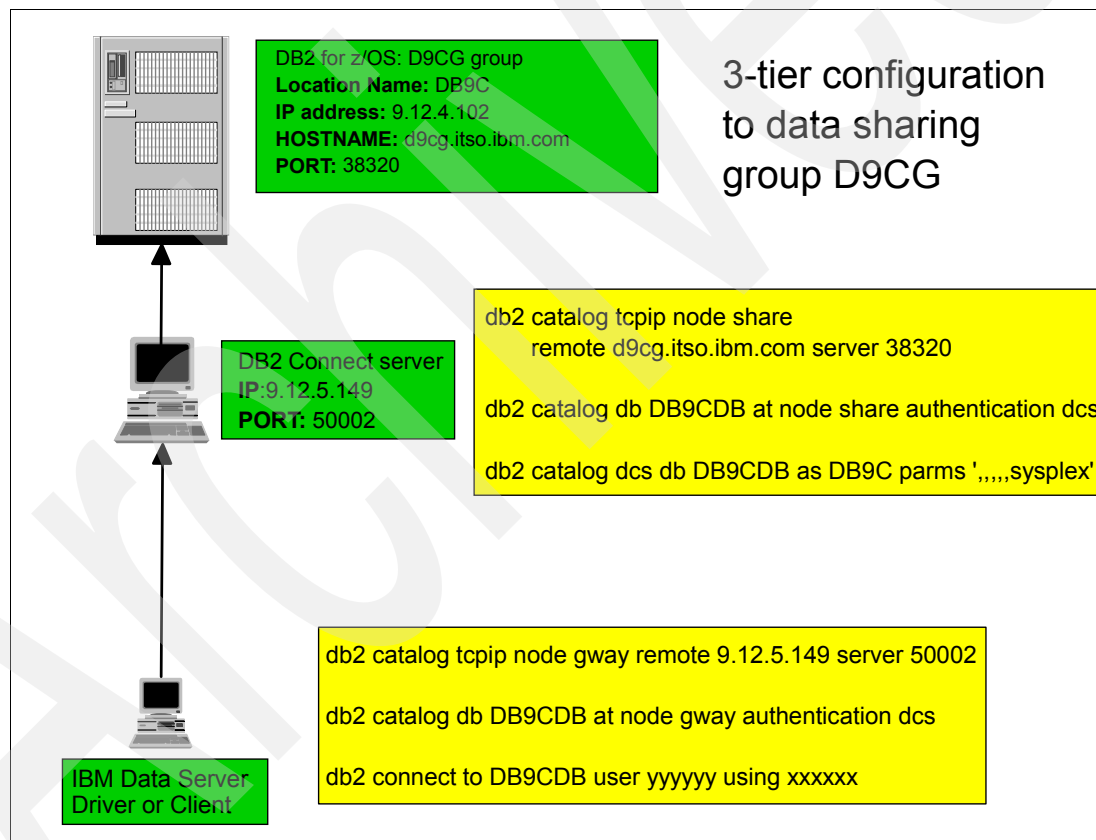


Figure 3-53 Three-tier connection to our DB2 for z/OS data sharing group

Table 3-9 on page 122 shows the commands on the left, and the information about the DB2 Connect server environment required on the right. The information about DB2 for z/OS data sharing group does not change, so the commands entered at the DB2 Connect server for the three-tier data sharing configuration are the same as those entered in the two-tier configuration in Table 3-8 on page 120.

Table 3-9 Commands to connect 3-tier to DB2 data sharing group and the information required

Commands to issue for workstation client	Information from DB2 Connect server environment (DB2 for z/OS does not change)
<b>db2 catalog tcpip node gway remote 9.12.5.149 server 50002</b> Notes: <ul style="list-style-type: none"> <li>▶ 'gway' is an arbitrary name for DB2 Connect server node (gateway).</li> <li>▶ We show the IP address could be used instead of host name</li> <li>▶ Specify the DRDA port number, or service name, after the 'server' keyword</li> </ul>	<ul style="list-style-type: none"> <li>▶ z/OS IP address) = 9.12.5.149</li> <li>▶ DRDA port number for DB2 Connect server = 50002</li> </ul>
<b>db2 catalog db DB9CDB at node gway authentication dcs</b> Notes: <ul style="list-style-type: none"> <li>▶ Other security options are available besides 'authentication dcs'</li> </ul>	<ul style="list-style-type: none"> <li>▶ DB9CDB is an arbitrary name you will use to connect to the DB2 for z/OS data sharing group.</li> </ul>
<b>db2 catalog dcs db DB9CDB as DB9C parms ',,,,,sysplex'</b> Note: <ul style="list-style-type: none"> <li>▶ parms ',,,,,sysplex' specifies sysplex support, including workload balancing</li> </ul>	<ul style="list-style-type: none"> <li>▶ LOCATION name for the DB2 for z/OS data sharing group = DB9C</li> </ul>
db2 connect to DB9CDB user yyyyyy using xxxxxx	<ul style="list-style-type: none"> <li>▶ user ID = yyyyyy ; password = xxxxxx</li> </ul>

## 3.5 DRDA sample setup—From DB2 for z/OS requester to DB2 for LUW on AIX server

This section provides the basic steps to connect DB2 for z/OS as a DRDA AR to a DB2 for LUW DRDA AS. The diagram and table that follow use the definition, names, and CDB values that you have seen earlier in this chapter.

**Important:** DRDA requires that a package be bound on the server. This is true no matter what platform is the requester nor what platform is the server.

Figure 3-54 on page 123 shows the steps and most of the information required to connect DB9A to the SAMPLE database in DB2 for LUW. This figure and Table 3-10 on page 123 assume that the DB2 for LUW database has been populated with the system and sample tables at least.

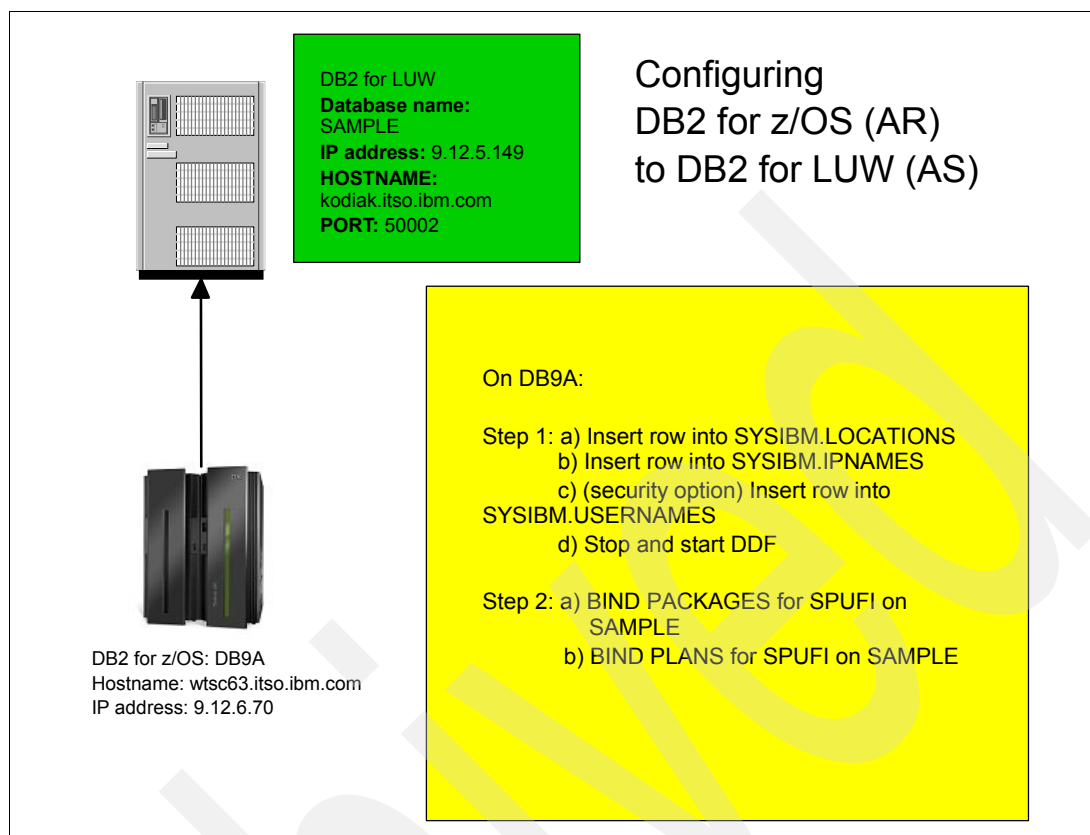


Figure 3-54 Steps to configure DB2 for z/OS as DRDA AR to DB2 for LUW as DRDA AS

In Table 3-10 we take each step and provide additional details, explanation and examples based on the configuration we used.

Refer to 3.2.2, “Configuring the Communications Database” on page 86 to see the corresponding CDB contents.

Table 3-10 Connecting DB2 for z/OS to DB2 for LUW

Steps to run on DB9A	Information needed from DB2 for LUW environment
<b>Step 1: Configure the Communications Database (CDB)</b>	
Insert into SYSIBM.LOCATIONS (location, linkname, port) values (‘SAMPLE’, ‘MYUDBLNK’, ‘50002’);	SAMPLE is the name of the DB2 for LUW database to which we want to connect.
Insert into SYSIBM.IPNAMES (linkname, security_out, usernames, ipaddr) values (‘MYUDBLNK’, ‘P’, ‘O’, ‘9.12.5.149’);	In our environment the IP address for DB2 for LUW is 9.12.5.149 and the port number is 50002. The DNS name of kodiak.itso.ibm.com could have been used, too.

Steps to run on DB9A	Information needed from DB2 for LUW environment
<p>Insert into SYSIBM.USERNAMES (type, authid, linkname, newauthid, password) values ('O', ' ', 'MYUDBLNK','DB2INST3', 'DB2INST3') ;</p> <p>Notes:</p> <ul style="list-style-type: none"> <li>► 'MYUDBLNK' is an arbitrary name to link SYSIBM.LOCATIONS, SYSIBM.IPNAMES and SYSIBM.USERNAMES.</li> <li>► The value 'P' for the SECURITY_OUT column means we must supply a password with a userid.</li> <li>► In our configuration we had a blank for the AUTHID column. We do not recommend leaving AUTHID blank in a production environment.</li> </ul>	<p>dbm cfg AUTHENTICATION should be set to SERVER, because we pass a user ID and password and expect DB2 for LUW to perform the authentication.</p> <p>As our CDB shows, we used the same user ID and password, DB2INST3.</p>
Stop and start DDF (-db9a stop ddf; -db9a start ddf) to ensure the changes take effect.	
<b>Step 2: Bind SPUFI</b>	
BIND PACKAGE (SAMPLE.DSNESPCS) MEMBER(DSNESM68) LIBRARY ('hlq.SDSNDBRM') ACTION (REPLACE) ISOLATION (CS) SQLERROR (NOPACKAGE) VALIDATE BIND	<p>SAMPLE is the DB2 for LUW database to which we want to connect</p> <p>This is the Cursor Stability SPUFI package.</p> <p>The user ID performing the bind should have been granted the appropriate privileges.</p>
BIND PACKAGE (SAMPLE.DSNESPRR) MEMBER(DSNESM68) LIBRARY ('hlq.SDSNDBRM') ACTION (REPLACE) ISOLATION (RR) SQLERROR (NOPACKAGE) VALIDATE (BIND)	<p>This is the Repeatable Read package.</p>
BIND PLAN (DSNESPCS) PKLIST (*.DSNESPCS.DSNESM68) ISOLATION (CS) ACTION (REPLACE)	<p>Using an asterisk (*) for the location in the PKLIST ensures the plan is bound in all locations.</p>
BIND PLAN (DSNESPRR) PKLIST (*.DSNESPRR.DSNESM68) ISOLATION (RR) ACTION (REPLACE)	

These steps should be enough to use SPUFI to test the connection by selecting from catalog tables on DB2 for LUW.

Repeat the package and plan binds for your application programs. Make sure both are bound at DB2 for z/OS and at DB2 for LUW. Ensure that appropriate security is in place before allowing access to production tables.



## 3.6 Character conversion: Unicode

When you transmit character data from one DBMS to another the data may need to be converted to a different coded character set. In different database management systems (DBMSs), character data can be represented by different encoding schemes. Within an encoding scheme, there are multiple coded character set identifiers (CCSIDs). EBCDIC, ASCII, and Unicode are ways of encoding character data. The Unicode character encoding standard is a character encoding scheme that includes characters from almost all languages of the world, including Latin. DB2 supports two implementations of the Unicode encoding scheme: UTF-8 (a mixed-byte form) and UTF-16 (a double-byte form).

All character data has a CCSID. Character conversion is described in terms of CCSIDs of the source and of the target. When you install DB2, you must specify a CCSID for DB2 character data in either of the following situations:

- ▶ You specify AUTO or COMMAND for the DDF STARTUP OPTION field on panel DSNTIPR.
- ▶ Your system will have any ASCII data, Unicode data, EBCDIC mixed character data, or EBCDIC graphic data. In this case, you must specify YES in the MIXED DATA field of panel DSNTIPF, and the CCSID that you specify is the mixed data CCSID for the encoding scheme.

The CCSID that you specify depends on the national language that you use.

DB2 performs most character conversion automatically, based on system CCSIDs, when data is sent to DB2 or when data is stored in DB2. If character conversion must occur, DB2 uses the following methods:

- ▶ DB2 searches the catalog table SYSIBM.SYSSTRINGS.
- ▶ DB2 uses z/OS Unicode Conversion Services.

If DB2 or z/OS Unicode Conversion Services does not provide a conversion for a certain combination of source and target CCSIDs, you receive an error message. If the conversion is incorrect, you might get an error message or unexpected output. In that case you may have to populate SYSIBM.SYSSTRINGS to specify the conversion you require.

Figure 3-55 on page 126 shows DB2 install panel DSNTIPF where you specify the CCSIDs for your system. Most data coming into your system through DRDA will be converted automatically by DB2 according to the principle “Receiver makes right”, which means the data recipient is responsible for the conversion.

```

DSNTIPF          INSTALL DB2 - APPLICATION PROGRAMMING DEFAULTS PANEL 1
====>

Enter data below:
1 LANGUAGE DEFAULT      ===> IBMCOB   ASM,C,CPP,IBMCOB,FORTRAN,PLI
2 DECIMAL POINT IS      ===> .         . or ,
3 STRING DELIMITER      ===> DEFAULT  DEFAULT, " or ' (IBMCOB only)
4 SQL STRING DELIMITER  ===> DEFAULT  DEFAULT, " or '
5 DIST SQL STR DELIMTR  ===> '         ' or "
6 MIXED DATA            ===> NO        NO or YES for mixed DBCS data
7 EBCDIC CCSID           ===>          CCSID of SBCS or mixed data. 1-65533.
8 ASCII CCSID            ===>          CCSID of SBCS or mixed data. 1-65533.
9 UNICODE CCSID          ===> 1208     CCSID of UNICODE UTF-8 data
10 DEF ENCODING SCHEME   ===> EBCDIC   EBCDIC, ASCII, or UNICODE
11 APPLICATION ENCODING  ===> EBCDIC   EBCDIC, ASCII, UNICODE, ccsid (1-65533)
12 LOCALE LC_CTYPE       ===>
13 DECFLOAT ROUNDING MODE===> ROUND_HALF_EVEN

```

Figure 3-55 DSNTIPF panel where you specify your system CCSIDs

One potential source of conversion errors is if a system on which you have bound a package has changed its system CCSID. Another conversion concern is adding support for the euro symbol. If you encounter conversion errors or need to support the euro symbol, refer to the *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846.

**Attention:** If you need to change the current CCSIDs for your system contact the IBM Support Center. Such a change is complex and you should follow the detailed plan that the IBM Support Center has developed for this change.

## 3.7 Restrictions on the use of local datetime formats

The following rules apply to the character string representation of dates and times:

► For input

In distributed operations, DB2 as a server uses its local date or time routine to evaluate host variables and literals. This means that character string representation of dates and times can be as follows:

- One of the standard formats.
- A format recognized by the server's local datetime exit.

► For output

With DRDA access, DB2 as a server returns date and time host variables in the format defined at the server.

► For BIND PACKAGE COPY

When binding a package using the COPY option, DB2 uses the ISO format for output values unless the SQL statement explicitly specifies a different format. Input values can be specified in the format described above under "For input".

## 3.8 HiperSockets: Definition

In this section we describe our HiperSocket definition. Configurations and considerations for the use of HiperSockets are described in 2.5, “DB2 Connect Server on Linux on IBM System z” on page 58.

HiperSockets represent the implementation of a virtual TCP/IP network path between LPARs. The links and addresses must be defined in the TCP profile member with DEVICE, LINK, HOME, ROUTE, and START entries.

Figure 3-56 shows the DEVICE, LINK and HOME entries. HiperSockets use TCP queued I/O support provided through IPAQIDIO. Our link is called HIPERLF1. The IP address for the HiperSocket on SC63 is 10.1.1.2.

```
BROWSE    TCP.SC63.TCPPARMS(TCPPROF) - 01.56
Command ==>

DEVICE OSA2000 MPCIPA
LINK   OSA2000LNK IPAQENET      OSA2000

DEVICE OSA2020 MPCIPA
LINK   OSA2020LNK IPAQENET      OSA2020

DEVICE IUTIQDF1 MPCIPA
LINK   HIPERLF1   IPAQIDIO      IUTIQDF1

HOME
  9.12.6.70      OSA2000LNK
  9.12.6.71      OSA2020LNK
  10.1.1.2      HIPERLF1
```

Figure 3-56 TCP profile extract with HiperSocket definitions - part 1

Figure 3-57 shows the remainder of the TCP profile member with the ROUTE entry for HIPERLF1 and the START entry to start the IUTIQDF1 adapter identified in the DEVICE and LINK statement.

```
BROWSE    TCP.SC63.TCPPARMS(TCPPROF) - 01.56
Command ==>

HOME
  9.12.6.70    OSA2000LNK
  9.12.6.71    OSA2020LNK
  10.1.1.2     HIPERLF1

ITRACE OFF

BEGINROUTES
ROUTE 9.12.4.0 255.255.252.0 = OSA2000LNK MTU 1500
ROUTE DEFAULT 9.12.4.1 OSA2000LNK MTU 1500
ROUTE 9.12.4.0 255.255.252.0 = OSA2020LNK MTU 1500
ROUTE DEFAULT 9.12.4.1 OSA2020LNK MTU 1500
ROUTE 10.1.1.0 255.255.255.0 = HIPERLF1 MTU 8192
ENDROUTES

START OSA2000
START OSA2020
START IUTIQDF1
```

Figure 3-57 TCP profile extract with HiperSocket definitions - part 2

For applications, including DB2 for LUW, on the Linux on z partition to reach SC63, they specify IP address 10.1.1.2.

# Security

The development of the Internet has dramatically changed the DRDA security landscape. The major environmental changes that have an impact on DRDA security are as follows:

- ▶ Application architectures have evolved, driven by the Internet. This has not only increased the need for security, but has also changed the nature of the security requirements for database access.
- ▶ SNA has largely been abandoned in favor of TCP/IP. This has removed a layer of security that was used extensively by DRDA configurations to provide security for links between DRDA client LUs and the APPLs that represent DB2 for z/OS.
- ▶ Many applications today use dynamic SQL. This generally requires explicit data access authorization, instead of just package/plan execution authorization.

The combined risks of TCP/IP sniffers, the growing use of dynamic SQL for ODBC and JDBC applications, and the growing use of group authorization IDs with substantial data access authorization, make protecting the connection authentication dialog more important than ever.

In this chapter we provide a brief summary of the basic security DRDA configuration over TCP/IP, explore the application server security options (including a mention of trusted context and roles), add considerations on encryption, and discuss the security issues with dynamic SQL.

This chapter contains the following sections:

- ▶ "Guidelines for basic DRDA security setup over TCP/IP" on page 130
- ▶ "DRDA security requirements for an application server" on page 140
- ▶ "Encryption options" on page 147
- ▶ "Addressing dynamic SQL security concerns" on page 173

## 4.1 Guidelines for basic DRDA security setup over TCP/IP

This section provides a brief overview of DRDA security over TCP/IP as a base for discussing the other security topics in the chapter. The basic facilities of DRDA security are authentication and authorization.

For details, refer to the following publications:

- ▶ *DB2 9 for z/OS Administration Guide*, SC26-9931
- ▶ *DB2 Connect Version 9 Quick Beginnings for DB2 Connect Servers*, GC10-4243
- ▶ *DB2 Connect Version 9 User's Guide*, SC10-4229

### 4.1.1 Security options supported by DRDA access to DB2 for z/OS

DRDA defines the information flows that allow a DRDA AR connection request to be authenticated at the DRDA AS.

The authentication mechanism to be used is specified on both the DRDA AR platform and the DRDA AS platform. Naturally, the DRDA AR platform cannot override or bypass the authentication standard demanded by the DRDA AS.

The security options supported by DB2 for z/OS and DRDA clients are listed in Table 4-1.

Table 4-1 Security options for DB2 for z/OS

Description	DB2 server support	DB2 Connect	Type 4 driver	CLI Driver	AES support
User ID password	Y	SERVER	CLEAR_TEXT_PASSWORD_SECURITY	SERVER	N
User ID only	Y	CLIENT	USER_ONLY_SECURITY	N	N
Change password	Y	N	N	N	N
User ID, password substitute	N	N	N	N	N
User ID, encrypted password	Y	N	ENCRYPTED_PASSWORD_SECURITY	N	Y
Encrypted user ID and password	Y	SERVER_ENCRYPT/ SERVER_ENCRYPT_AES	ENCRYPTED_USER_AND_PASSWORD_SECURITY	SERVER_ENCRYPT/ SERVER_ENCRYPT_AES	Y
Encrypted change password	Y	N	N	N	Y
Kerberos	Y	KERBEROS	KERBEROS_SECURITY	KERBEROS	N

Description	DB2 server support	DB2 Connect	Type 4 driver	CLI Driver	AES support
Encrypted user ID and data	Y	N	ENCRYPTED_USER_AND_DATA_SECURITY	N	N
Encrypted user ID, password, and data	Y	DATA_ENCRYPT	ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY	DATA_ENCRYPT	N
Encrypted user ID, password, new password, and data	Y	N	N	N	N
Encrypted user ID only	Y	N	ENCRYPTED_USER_ONLY_SECURITY	N	Y
SSL	Y	Y	Y	Y	N/A

For example, the authentication mechanism to be used by DB2 Connect is specified on the catalog database command, as shown in Example 4-1. Similar settings apply also to T4 Driver or CLI driver. Some examples are given later in the chapter.

*Example 4-1 Authentication mechanism by catalog database command*

```
$ db2 catalog db DB9C at node WTSC63 authentication SERVER_ENCRYPT_AES
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
```

The processing flow of inbound DRDA connections over TCP/IP to DB2 for z/OS is shown in Figure 4-1 on page 132. The process can be described as follows:

1. Is the authentication information present?

The only case where authentication information is not required is when TCPALVER=YES is specified for the subsystem. Setting TCPALVER=YES is not a recommended option. Review the information in 4.1.3, "Important considerations when setting security related DSNZPARMs" on page 132, before you set your parameter.

2. Check the identity of the ID.

A RACF ID (or equivalent security software product) must exist in all cases. The RACF ID is either a RACF USER ID or a RACF ID that is derived from the Kerberos principal identity.

3. Check that the ID is allowed to connect to DB2 for z/OS through the DDF address space.

Verify that the RACF ID is permitted to access the ssid.DIST profile in the DSNR resource class for the target DB2 subsystem (ssid).

To permit access to D9C1 through DRDA to user ID DB2USER1, issue the following command:

```
PERMIT D9C1.DIST CLASS(DSNR) ID(DB2USER1) ACCESS(READ)
```

4. Run the DB2 connection exit.

The connection exit routine(DSN3@ATH) can be used to perform a variety of functions. It is usually used to assign secondary authorization IDs based on the list of RACF groups a user belongs to. TCP/IP requests do not use the sign-on exit routine (DSN3@SGN).

5. Local authorization checking.

Validate that the primary authorization ID (PAI) or any secondary authorization ID (SAI) has the authorization to execute the specified package and access to the DB2 resources referenced in the SQL statement.

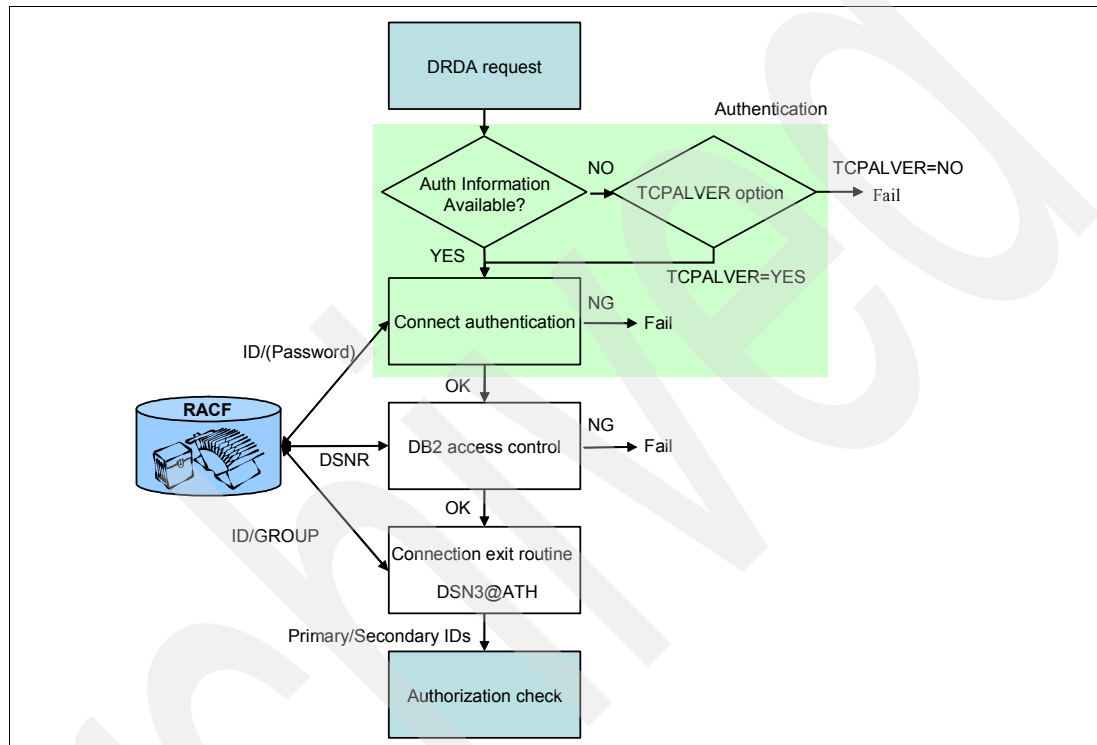


Figure 4-1 DRDA connection flow of DB2 for z/OS

## 4.1.2 Authorization

Once having gone through DRDA authentication processing, authorization verification is handled in an identical way as for local processing.

## 4.1.3 Important considerations when setting security related DSNZPARMs

Considering the importance of your enterprise data, you must evaluate the following two DSNZPARMs before allowing use of DB2 data through DRDA.

- ▶ TCPALVER
- ▶ EXTSEC

### Setting the TCPALVER parameter

TCPALVER=NO (on DSNZPARM panel DSNTIP5) is strongly recommended for most TCP/IP networks because of the potential security exposure from unplanned DRDA AR connections.



If you specify TCPALVER=YES, RACF will not perform password checking unless the connection request sends the password. DRDA ARs can force a password to be sent by specifying one of server authentication attributes.

If you want to use TCPALVER=YES (to avoid RACF password checks, since the user ID is still verified against RACF), be aware of the security risks. Evaluate the following issues when setting TCPALVER.

- ▶ Is the privilege to set authentication attribute to DRDA AR restricted?

If a person has authority to change the authentication attribute of DRDA AR clients, this person can change the attribute to CLIENT authentication and allow access to DB2 for z/OS without passwords.

- ▶ Is the workstation restricted to use external media?

If a user can install any DRDA AR client (like DB2 Connect Personal Edition), this user may catalog a link to DB2 for z/OS with authentication CLIENT to access DB2 for z/OS without passwords.

- ▶ Is the network physically separated and secured?

If a person can bring a notebook computer to simply plug into a network and have access to DB2 for z/OS. Having a DRDA AR client (such as DB2 Connect Personal Edition) installed in the notebook computer likely provides the privilege to use or change the DRDA AR with authentication CLIENT.

### ***Enhancements to TCPALVER parameter***

The DRDA technology opened doors to accessing mainframe data from the intranet or Internet. This also opens doors to ill-intentions and vulnerability. Many countries passed laws related to IT security, and companies are implementing tighter corporate security policies to comply with them.

For DRDA, most of the authentication mechanisms can be chosen at the DRDA AR sites, where many users have the authority to access those settings. It is not hard to change those settings causing security vulnerability. And it is hard to check whether all clients connecting to DB2 for z/OS are fully complying with the policies when hundreds of clients are connecting to the server through the network.

With APAR PK76143, DB2 for z/OS has introduced the new security validation option SERVER\_ENCRYPT for the TCPALVER parameter to solve this issue. The previous options were YES to indicate that remote clients can access DB2 for z/OS without password, and NO (default), which indicates that DB2 for z/OS requires a password authentication whether encrypted or not.

**Attention:** This function was not tested since APAR PK76143 was still open at the time of writing.

The new SERVER\_ENCRYPT value enforces clients to make encrypted authentication. All the DRDA AR clients need to authenticate with DB2 for z/OS with one of following authentication mechanism:

- ▶ Password encryption (AES encryption only)
- ▶ SSL connection (also AT-TLS)
- ▶ IP Security (IPSec)
- ▶ Kerberos

After specifying SERVER\_ENCRYPT, if the DRDA AR clients attempts to connect using an unsupported security mechanism, DB2 for z/OS will reject the connection and issue the DSNL030I message with reason code 00D31050.

In addition, the specification of options SERVER and CLIENT has been added for the TCPALVER DSNZPARM for reasons of DB2 family compatibility. The SERVER value is an alternative to the NO option and the CLIENT value is an alternative value to the YES option. This makes it easier to match server with client settings.

**Note:** Although DB2 for z/OS still supports 56-bit DES encryption, it is now considered an insecure form of authentication. The new feature will not support 56-bit DES encryption as an acceptable security mechanism.

RACF PassTickets is not included in this support, since the password is only encoded and not as strong as with DES- or AES-based encryption. In addition, users with a high transaction rate typically define RACF PassTickets as replay capable, and this weakens the security protection further. PassTickets will only be accepted if the connection itself is protected (by AT-TLS or IPSec).

**Note:** When using AT-TLS or IPSec, DB2 will validate the presence of an IPSec encryption tunnel. Because of this you will need to pay the expense to ensure you are accessing DB2 for z/OS in clear text but using IPSec. The IPSec validation only occurs for initial requests.

## Extended security

The extended security option in DSNZPARM panel DSNTIPR is a must have option. It provides two key functions:

- ▶ When a user or program is rejected by the authentication processing, the specific reason for rejection is returned in the SQLCA. Example 4-2 shows the results of a connection request with an invalid password, without extended security, and then with the extended security. With extended security, the user can easily determine the reasons for failure.

*Example 4-2 Authentication rejection with and without extended security*

---

### Connection request with invalid password, with EXTSEC=NO

```
$ db2 connect to db9a user paolor7 using password
SQL30082N Security processing failed with reason "15" ("PROCESSING FAILURE").
SQLSTATE=08001
```

### Connection request with invalid password, with EXTSEC=YES

```
$ db2 connect to db9a user paolor7 using password
SQL30082N Security processing failed with reason "24" ("USERNAME AND/OR
PASSWORD INVALID"). SQLSTATE=08001
```

**In both cases, SYSLOG output explains that the user has given invalid password**

```
ICH408I USER(PAOLOR7 ) GROUP(SYS1 ) NAME(P. ) 663
LOGON/JOB INITIATION - INVALID PASSWORD
IRR013I VERIFICATION FAILED. INVALID PASSWORD GIVEN.
DSNL030I -DB9A DSNLTSEC DDF PROCESSING FAILURE FOR 665
LUWID=G90C0595.B75C.C400E544C97B
AUTHID=paolor7, REASON=00F30085
```

---

- The second function of extended security is the change password API. This allows the RACF password to be changed as part of a DRDA AR (like DB2 Connect) request, as shown in Example 4-3.

*Example 4-3 Change RACF password on connect*

---

```
$ db2 connect to db9a user paolor7 using passwd new newpswd confirm newpswd
```

---

Database Connection Information

Database server	= DB2 z/OS 9.1.5
SQL authorization ID	= PAOLOR7
Local database alias	= DB9A

---

With the increasing deployment of distributed applications, where users do not necessarily log on to TSO, or even have TSO access disabled from their RACF profile, the ability to change passwords from the client environment, or from within a program, can be useful. The change password API is available in all the programming languages supported by the DB2 client.

#### 4.1.4 Recommendation for tightest security

For the tightest security, do not send a clear text password through the network. Instead, consider using one of the following security options:

- RACF PassTicket
- Kerberos ticket
- DRDA encrypted passwords

##### RACF PassTicket

RACF PassTicket is a cryptographically generated short life-span alternative to the RACF password that can be used to get authenticated to other DB2 for z/OS, as illustrated in Figure 4-2 on page 136. This means DB2 for z/OS DRDA AR does not have to send RACF passwords in clear text through a network.

**Note:** As described in the previous section, RACF PassTickets is not allowed with the new SERVER\_ENCRYPT specification for TCPALVER, which restricts the security mechanism on the server.

RACF PassTicket is not a replacement of the regular RACF password, which remains usable. DB2 for z/OS DRDA AR users can use RACF PassTicket as substitute to RACF passwords. This avoids the need to code passwords in the CDB (SYSIBM.USERNAMES) and update them each time a password expires. Unlike the RACF password, the RACF PassTicket applies to only one applications, and remains valid for a period of about 10 minutes, which makes it easier to manage and keeps it secure.

**Note:** You can also use RACF PassTicket on distributed platform using Tivoli® Federated Identity Manager (TFIM), sending request to DB2 for z/OS without needs to code a password on your clients. Refer to TFIM product documentation for details on using RACF Passticket on distributed platforms.

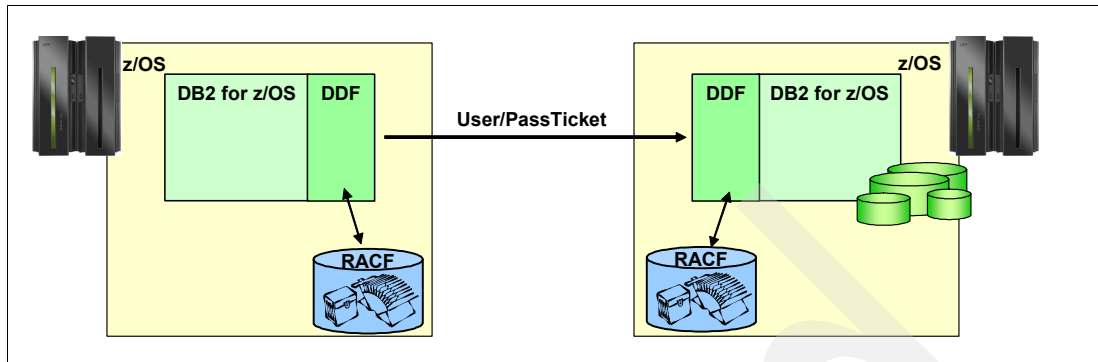


Figure 4-2 Connecting to DB2 using RACF PassTickets

RACF PassTicket is generated from the following attributes:

- ▶ The user ID of the client
- ▶ The application ID (CICS applid, IMS id, ...)
- ▶ The LINKNAME in the requester CDB, and the server LUNAME (non-data sharing), or generic LUNAME (data sharing), or IPNAME (TCP/IP with VTAM independence)
- ▶ A secured sign-on application key, known to both sides of the connection
- ▶ A time and date stamp

**Restriction:** A RACF PassTicket is valid only once. The same application generates only one different PassTicket every few seconds. If fast requests are made, the same PassTicket can be used, which causes denied access. In such an environment, the NO REPLAY PROTECTION text string in the APPLDATA field of the PTKTDATA RACF general resource class profile will bypass PassTicket replay protection. Before using this option, ensure it is only used in secured environments, such as limited access or physically separated network environments.

### implementing RACF PassTicket

Perform the following steps to implement RACF PassTicket.

1. Set the SECURITY\_OUT column of the SYSIBM.SYSIPNAMES to R.
2. Define RACF resource at requesting system

Activate the RACF PTKTDATA class by using the RACF command shown in Example 4-4.

*Example 4-4 Activates PTKTDATA class*

---

```
SETROPTS CLASSACT(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)
```

---

Define a profile for each DB2 to which the application is connecting. This means that profiles need to be defined for each LINKNAME columns of SYSIBM.LOCATION. KEY\_MASKED\_VALUE is security sensitive information given when profiles are defined. When the DB2 subsystem you are connecting to is using a different RACF database, classes and profiles are required to be defined with the same KEY\_MASKED\_VALUE.

For example, if you are connecting to DB9A using KEY\_MASKED\_VALUE of E001193519561977 with the command shown in Example 4-5.

*Example 4-5 New profiles to remote DB2 subsystem*

---

```
RDEFINE PTKTDATA DB9A SSIGNON(KEYMASKED(E001193519561977))
SETROPTS RACLIST(PTKTDATA) REFRESH
```

---

### 3. Define RACF resources at remote system.

As mentioned, if the server RACF database is different from the requesting DB2, you must define classes and profiles using the same KEY\_MASKED\_VALUE.

## Kerberos ticket

One of the most secure form of authentication encryption for DRDA connections to DB2 for z/OS is to use the Kerberos authentication mechanism. Kerberos support was not tested during the writing of this publication, so this section is limited to a theoretical discussion of what should be done.

Kerberos derives its name from Cerberus, the mythological three-headed dog that guarded the entrance to the underworld. Kerberos was developed by MIT as a distributed authentication service for use over an untrusted network. Kerberos acts as a trusted third party, responsible for issuing user credentials and tickets.

Users and servers are required to have keys registered with the Kerberos authentication server. When a user wants to access a server, the Key Distribution Centre (KDC) issues a ticket for access to the server. Tickets contain a client's identity, a dynamically created session key, a time stamp, a ticket lifetime, and a service name.

The KDC consists of the following elements:

- ▶ A Kerberos authentication server (KAS)
- ▶ A ticket granting server (TGS)
- ▶ A Kerberos database (KDB)

Kerberos is implemented in the following component of z/OS SecureWay™ Security Server:

- ▶ z/OS SecureWay Security Server Network Authentication and Privacy Service
- ▶ RACF or equivalent security product

For detailed information about setting up a Kerberos authentication service on z/OS, refer to *z/OS V1R10.0 Network Authentication Service Administration*, SC24-5926.

**Note:** DB2 for z/OS (as a DRDA AS) can only verify a Kerberos ticket when it is being passed. DB2 for z/OS (as a DRDA AR) does not request authentication tickets from a Kerberos server.

Regarding the DRDA usage of Kerberos, the process is similar to other forms of authentication. How to catalog a database with Kerberos principal server authentication to DB2 Connect is shown in Example 4-6. Other DRDA AR Clients have options to choose Kerberos authentication.

*Example 4-6 Catalog a database using Kerberos authentication*

---

```
catalog database <dbname> as <dbalias> at node <node> authentication kerberos
target principal
```

---

The authentication flow using Kerberos connecting to DB2 for z/OS is illustrated in Figure 4-3.

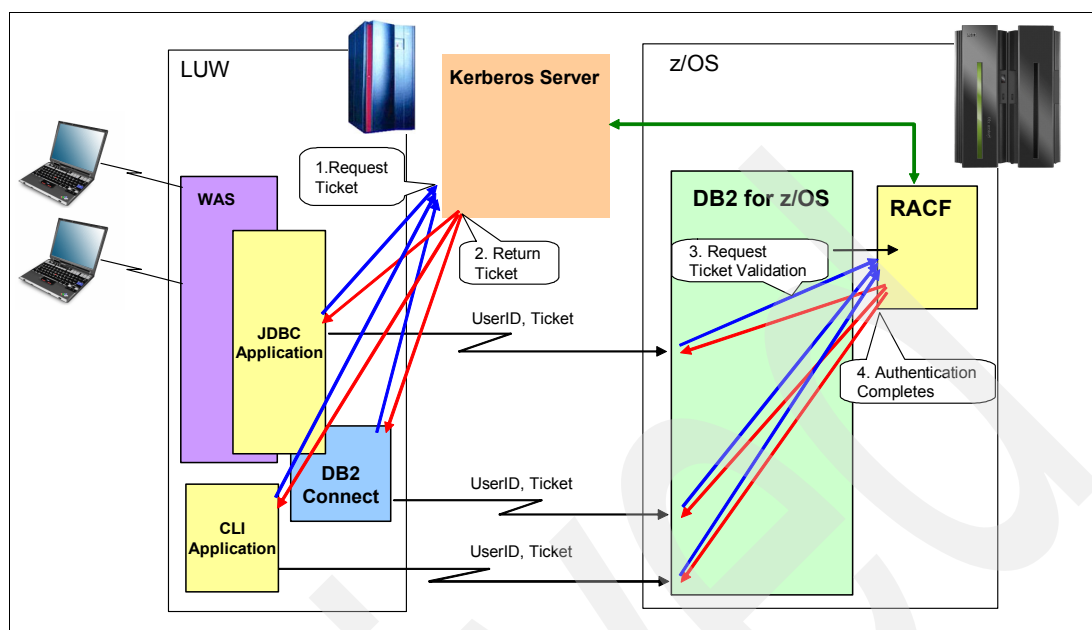


Figure 4-3 Authentication process using Kerberos

## DRDA encrypted passwords

Depending on the DRDA level, DRDA AR can encrypt one of the following when you send them to DB2 for z/OS server.

- ▶ Password
- ▶ User ID and password
- ▶ User ID, password and security sensitive data

In this section we show examples of setting encrypted password. In DB2 9 for z/OS, advanced options like AES encryption or SSL are provided to offer tighter security, those options are discussed in 4.3, “Encryption options” on page 147.

## Setting distributed clients to use encrypted passwords

As described in 4.1.1, “Security options supported by DRDA access to DB2 for z/OS” on page 130, you can set the supported authentication mechanism to encrypt your password flow through the network. If one of the previous options are not being used, we recommend using one of the options to encrypt your password.

Example 4-7 shows the setting to use encrypted user ID and password using DB2 Connect.

### Example 4-7 Example for setting authentication mechanism at DB2 Connect

```
$ db2 catalog db DB9A at node SC63TS authentication SERVER_ENCRYPT
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
```

In WebSphere Application Server environments, you can change the connection properties to use different security options. For example, if you want to use encrypted user ID and password, you can set your properties as shown in Figure 4-4 on page 139.

<input type="checkbox"/>	<a href="#">securityMechanism</a>	<a href="#">9</a>	<a href="#">Specifies the DRDA security mechanism. Possible values are: 3 (CLEAR TEXT PASSWORD SECURITY), 4 (USER ONLY SECURITY), 7 (ENCRYPTED PASSWORD SECURITY), 9 (ENCRYPTED USER AND PASSWORD SECURITY), or 11 (KERBEROS SECURITY). Please refer to DB2 manual for the constant value. If this property is specified, the specified security mechanism is the only mechanism that is used. If no value is specified for this property, 3 is used.</a>	<a href="#">false</a>
--------------------------	-----------------------------------	-------------------	---	-----------------------

Figure 4-4 Setting for the DataSource custom properties on WebSphere Application Server

### Setting DB2 for z/OS DRDA AR to use encrypted passwords

If you are connecting from DB2 for z/OS (as DRDA AR), you need to populate the valid user ID and password in the communication database (CDB) tables to connect to remote DB2 for z/OS server. If you are using INSERT statements to populate the CDB tables, you need to set your user IDs and password in clear text. If the network is connected to the Internet or a wide-range of intranets this can be a security exposure.

DB2 for z/OS provides the DSNLEUSR stored procedures to let users store encrypted translated authorization ID (NEAUTHID) and password (PASSWORD) in the SYSIBM.USERNAMES table.

**Note:** To use the DSNLEUSR stored procedures, the following conditions must be met:

- ▶ DB2 for z/OS V8 or later.
- ▶ WLM established stored address space.
- ▶ z/OS Integrated Cryptographic Service Facility (ICSF) must be installed, configured, and active.

The DB2 manual, *DB2 9 for z/OS Administration Guide*, SC18-9840, provides a good example of executing the DSNLEUSR stored procedure from a COBOL program. From an administration and security point of view, you should restrict access to the DSNLEUSR stored procedure to security or database administrators.

Figure 4-5 shows the syntax for the DSNLEUSR stored procedure.

```
>>__CALL__DSNLEUSR__(__Type,__AuthID__,__LinkName__,__NewAuthID__,>
|_NULL_|_|_NULL_|_|_NULL_|
>__Password__,__ReturnCode__,__MsgArea__)><
|_NULL_|
```

Figure 4-5 Syntax diagram for the DSNLEUSR

Example 4-8 shows how to invoke the stored procedure from DB2 Connect. The procedure asks for five input parameters and two output parameters (to return SQLCODE and message.)

#### Example 4-8 Example of executing SYSPROC.DSNLEUSR(from DB2 Connect)

```
$ db2 "call SYSPROC.DSNLEUSR('0','PAOLOR7','DB9A','PAOLOR7','NEWPSWD',?,?)"
RETURNCODE: 0
MSGAREA: INSERTED INTO SYSIBM.USERNAMES SUCCESSFULLY.
```

```
"DSNLEUSR" RETURN_STATUS: 0
```

Example 4-9 shows the inserted translated authorization ID and password from the SYSIBM.USERNAMES CDB table. Unlike distributed DRDA AR clients, when encrypted authorization ID and password are inserted in CDB table, DB2 for z/OS automatically chooses the security mechanism, so you only need to set it to 'O'.

*Example 4-9 Display of inserted row of SYSPROC.DSNLEUSR*

LINKNAME	NEWAUTHID	PASSWORD
DB9A	..ba0c88441ef3b6e9	..939ce233c947f266
DSNE610I NUMBER OF ROWS DISPLAYED IS 1		

**Note:** There are no way to tell the value of encrypted translated authorization IDs or encrypted passwords once they are inserted in CDB table. If you need to update a password, you need to DELETE the current row before executing the DSNLEUSR stored procedure again.

## 4.2 DRDA security requirements for an application server

With the growing use of the application server model, it is generally accepted that the security challenges, authentication, and policy enforcement are increasingly being coordinated from within the application server environment, and DB2 accepts DRDA requests from application server threads, which act on behalf of the real users.

### 4.2.1 Characteristics of a typical application server security model

The security controls in an application server environment are typically enforced when a user signs in or requests access to a function that requires authentication. The user will be authenticated with a user ID that is probably not defined to RACF. The database access is usually performed under the generic user ID of the application server.

There are a number of reasons for this approach:

- ▶ It enables application connection pooling to be used.
- ▶ It does not externalize user IDs with database privileges to the Internet.
- ▶ It makes the user ID administration easy to manage.

Figure 4-6 on page 141 shows a typical Web application server security model, where clients are authenticated at the Web application server and a generic user ID is used to connect to DB2.



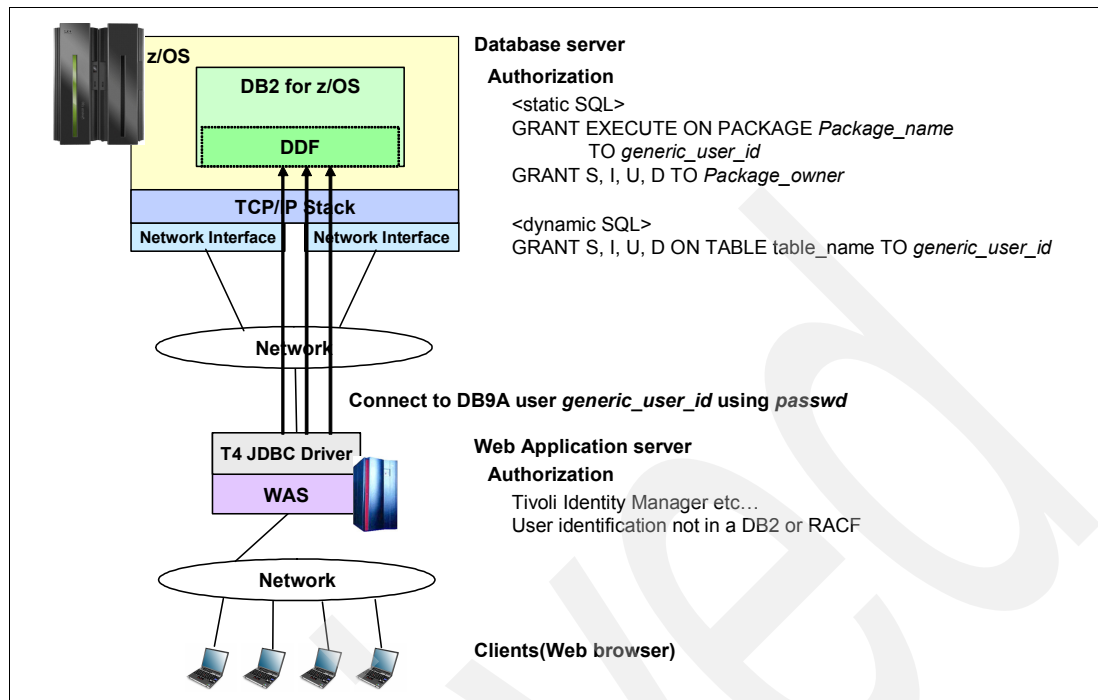


Figure 4-6 Typical DRDA access Web application server security model

The diagram does not attempt to illustrate how firewalls would be set up to establish a demilitarized zone (DMZ)<sup>1</sup>, since that is outside the scope of this publication. Suffice to say that the firewall configuration would permit the TCP/IP connections illustrated in the diagram, and nothing else.

The fact that the user authentication challenge is handled before a DB2 request is made does not alleviate DB2 from security considerations. It just changes the security exposures that DB2 needs to be concerned with.

## 4.2.2 Considerations for DRDA security behind the application server

Given the typical application server security model, the major considerations for DB2 security in this environment are:

- ▶ **Firewalls**  
 To keep the bad guys out (assuming that they are outside of DMZ.)
- ▶ **Encryption**  
 Encryption of authentication flows to provide protection against any bad guys who are already inside the DMZ.
- ▶ **Static SQL security**  
 Whenever possible, granting access to an executable package is preferable to granting access to the underlying table or view.

TCP/IP security is outside the scope of this publication, but the topics of encryption and SQL security are discussed.

<sup>1</sup> Servers in the DMZ provide services to both the internal and external network, while an intervening firewall controls the traffic between the DMZ servers and the internal network clients.

### 4.2.3 Identifying a client user coming from the application server

A common requirement for identifying a user is to determine who is having or causing a problem, for accounting or for auditing. As explained in 4.2.1, “Characteristics of a typical application server security model” on page 140, there are some good reasons to use generic user ID for the application server. However, a generic user ID makes it difficult to determine who is having problems using tools like IBM Tivoli OMEGAMON® XE for DB2 Performance Expert (OMEGAMON PE)<sup>2</sup>. If Accounting and Audit logs are populated with generic user ID and java process, it is impossible to determine who is doing what.

DB2 provides special registers where to put client information, and DRDA provides a solution by providing a way to pass client information in the DRDA data flow. Depending on the available API, you will use two different methods to set these fields.

- ▶ The data server driver (ODBC/CLI/.NET)
- ▶ The T4 driver (JDBC)

#### Set client information for an application server level

The client informations can be populated by setting properties of application server of datasource, which will be passed to DB2 for z/OS as connection attributes. If you have several Application Servers, each having different applications, then you change the setting for each application server or a datasource to determine applications having different datasource. Figure 4-7 shows the possible client settings. For more granular determination or auditing, you need to set client information about each applications.

<input type="checkbox"/>	<a href="#">clientAccountingInformation</a>		<a href="#">Specifies accounting information for the current client for the connection. This information is for client accounting purposes. This value can change during a connection. For a DB2 UDB for Linux, UNIX and Windows server, the maximum length is 255 bytes. A Java empty string is valid for this value, but a Java null value is not valid.</a>	false
<input type="checkbox"/>	<a href="#">clientApplicationInformation</a>		<a href="#">Specifies application information for the current client for the connection. This information is for client accounting purposes. This value can change during a connection. For a DB2 UDB for Linux, UNIX and Windows server, the maximum length is 255 bytes. A Java empty string is valid for this value, but a Java null value is not valid.</a>	false
<input type="checkbox"/>	<a href="#">clientUser</a>		<a href="#">Specifies the current client user name for the connection. This information is for client accounting purposes. Unlike the JDBC connection user name, this value can change during a connection. For a DB2 UDB for Linux, UNIX and Windows server, the maximum length is 255 bytes.</a>	false
<input type="checkbox"/>	<a href="#">clientWorkstation</a>		<a href="#">Specifies the workstation name for the current client for the connection. This information is for client accounting purposes. This value can change during a connection. For a DB2 UDB for Linux, UNIX and Windows server, the maximum length is 255 bytes. A Java empty string is valid for this value, but a Java null value is not valid.</a>	false

Figure 4-7 Client Informations settings through WebSphere Application Server admin console

For those who are using ODBC/CLI/.NET with CLI driver or .NET driver, you can also set client information at either db2cli.ini or db2dsdriver.cfg configuration files to set a value at driver installation level, in addition to connection attributes. Example 4-10 on page 143 shows a sample configuration file which sets ClientWorkstationName to value “my workstation name”.

<sup>2</sup> IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, V4.2 has been recently announced, see: [http://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&apname=iSource&supplier=897&letternum=ENUS209-077&open&cm\\_mmc=4895-\\_-n-\\_-vrn\\_newsletter-\\_-10207\\_114312&cmibm\\_em=dm:0:16594050](http://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&apname=iSource&supplier=897&letternum=ENUS209-077&open&cm_mmc=4895-_-n-_-vrn_newsletter-_-10207_114312&cmibm_em=dm:0:16594050)

The following list shows the IBM Data Server Driver configuration keywords:

- ▶ ClientUserID
- ▶ ClientApplicationName
- ▶ ClientWorkstationName
- ▶ ClientAccountingString

**Note:** If you have installed one of DB2 Connect, Clients, or runtime, db2dsdriver.cfg will not be used by the CLI driver. In most of cases, your application server should be able to provide you with set connection attributes but if that is not the case, use db2cli.ini.

*Example 4-10 Sample configuration file contents of data server driver*

```
<configuration>
  <DSN_Collection>
    <dsn alias="DB9C" name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="Authentication" value="Server_encrypt"/>
    </dsn>
  </DSN_Collection>
  <databases>
    <database name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="ClientWorkstationName" value="my workstation name"/>
    </database>
  </databases>
  <parameters>
    <parameter name="CommProtocol" value="TCPIP"/>
  </parameters>
</configuration>
```

If you are using ODBC driver in Windows environment, Figure 4-8 gives a sample configuration to set your client information from your Data Source ODBC control panel window. Panel will populate your db2cli.ini configuration file.

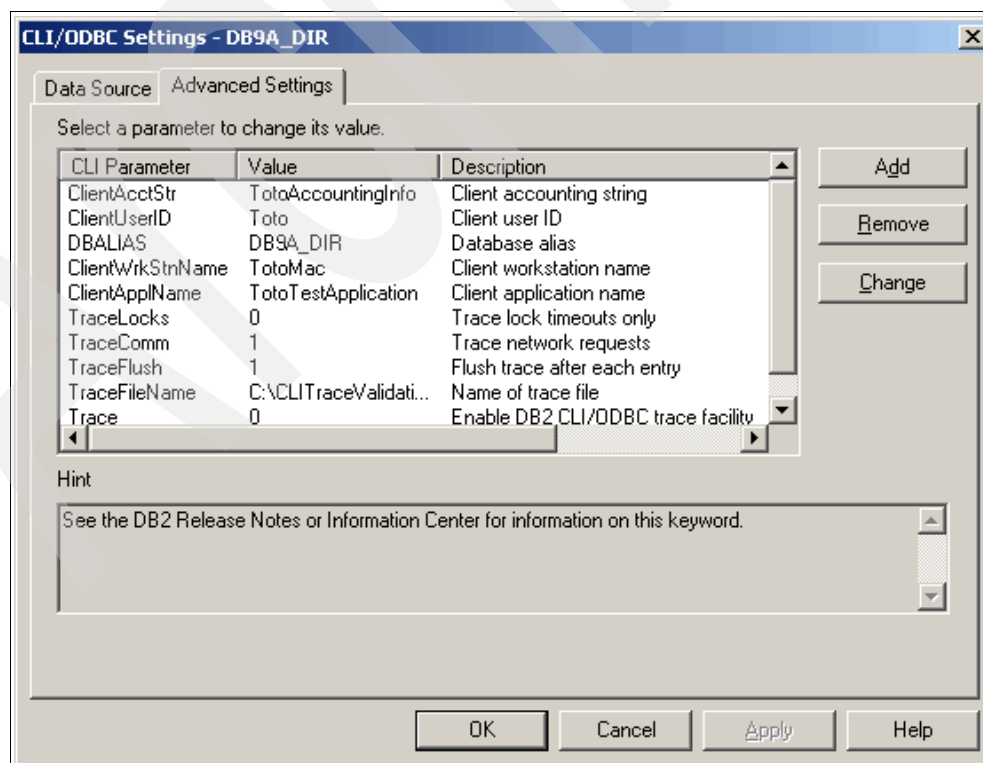


Figure 4-8 Client information setting example using datasource ODBC settings

**Tip:** There are several ways of adding entries into your db2cli.ini configuration file, such as datasource ODBC settings, CLI panel, or even a text editor.

## Set client information for applications

The application server may use a generic user ID to connect to DB2 for z/OS on behalf of client users. In this case, if you need to audit data accesses, you need to set the client information through properties or API before the DB2 transaction starts authentication at the application server. DRDA AR clients provide sets of API to set client information from each application.

**Tip:** When setting client information in the application, be sure to include them in your application standards.

Here we provide examples of code for various applications to show how client information can be populated from the applications.

Example 4-11 shows a Java application example of setting client information.

*Example 4-11 Sample for setting client information from Java applications*

---

### Client information set from connection properties.

```
java.util.Properties properties = new java.util.Properties();
    properties.put("clientAccountingInformation", "PAYROLL");
c = java.sql.DriverManager.getConnection (url, properties);
```

### Client information set from methods provided by T4 Driver

```
(com.ibm.db2.jcc.DB2Connection)db2conn.setDB2ClientUser(c1_user_name);
(com.ibm.db2.jcc.DB2Connection)db2conn.setDB2ClientWorkstation(c1_ws_name);
(com.ibm.db2.jcc.DB2Connection)db2conn.setDB2ClientApplicationInformation(c1_app);
(com.ibm.db2.jcc.DB2Connection)db2conn.setDB2ClientAccountingInformation("PAYROLL"
);
```

---

Example 4-12 shows an example of setting client information at WebSphere Application Server application using WSCConnection class.

*Example 4-12 Sample setting client information from WebSphere Application Server applications*

---

```
WSCConnection conn = (WSCConnection) ds.getConnection();
Properties props = new Properties();
props.setProperty(WSCConnection.CLIENT_ID, c1_user_name);
props.setProperty(WSCConnection.CLIENT_LOCATION ,c1_ws_name);
props.setProperty(WSCConnection.CLIENT_APPLICATION_NAME,c1_app);
...
conn.setClientInformation(props);
```

---

Example 4-13 shows an example of setting client information from ODBC/CLI applications.

*Example 4-13 Sample setting client information in ODBC/CLI applications*

---

**Information can be set as connection attribute using clientid char variable**

```
RETCODE = SQLSetConnectAttr(hDbc, SQL_ATTR_INFO_USERID,  
(SQLPOINTER)clientid,SQL_NTS);  
if(RETCODE != SQL_SUCCESS){  
    printf("allocate conn attr unsuccessful. \n");  
    return(-1);  
}
```

**Informations can be set within application using API**

```
sqlseti(dbAliasLen, dbAlias, 1, &clientAppInfo[0], &sqlca);
```

---

If you are writing a Visual Basic® application, you can use DB2 .NET Data provider. It provides a set of API, and, like for the other drivers, you can set your properties of DB2Connection class, as shown in Example 4-14.

*Example 4-14 Sample setting client information in ADO.NET application (Visual Basic)*

---

```
con = New DB2Connection(myConnString)  
con.ClientUser = cl_user_name  
con.Open()
```

---

**Note:** If you are setting client information from your application, you should not set any client information from any configuration file.

## 4.2.4 Network trusted context and roles

DB2 9 introduced new options for tighter security allowing more granularity and additional flexibility. These options are implemented to DB2 for z/OS by the following two new entities:

- ▶ Trusted context
- ▶ Role

A trusted context establishes a trusted relationship between DB2 and an external entity, such as middleware server or another DB2 subsystem. At connect time, sets of trusted attributes are evaluated to determine if a specific context can be trusted. After the trusted connection is established, sets of privileges and roles will be assigned to give access to DB2 data. Roles are not available outside of the trusted connection.

When defined, connections from specific users through source servers allow trusted connections to DB2. The users in this context can also be defined to obtain a database role.

A role is a database entity that groups together one or more privileges and can be assigned to users. A role can provide privileges that are in addition to the current set of privileges granted to the user's primary and secondary authorization IDs.

Users must be allowed to use a trusted context. A trusted context can exist without a role. A role is usable only within an established trusted connection. A default role can be assigned to a trusted context.

Within a trusted connection, DB2 allows one and only one role to be associated with a thread at any point in time.

For detailed explanation and examples using trusted context and roles, see *Securing DB2 and Implementing MLS on z/OS*, SG24-6480 and the article “End-to-end federated trusted contexts in WebSphere Federation Server V9.5”, available from the following Web page:

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0712baliga/index.html>

Additionally, we briefly discuss here new functions introduced by DB2 9 APAR PK44617. The functions added in this APAR are as follows:

- ▶ EXTERNAL SECURITY PROFILE profile-name
- ▶ Attribute JOBNAME with wildcard character '\*\*'
- ▶ The ROLE AS OBJECT OWNER clause is modified as ROLE AS OBJECT OWNER AND QUALIFIER to indicate that the role value is used as the default for CURRENT SCHEMA and CURRENT PATH special registers, as well as the default qualifier for the EXPLAIN tables.

Before this change, the users allowed to switch within a trusted context could not be defined in RACF. This makes it difficult for security (RACF) administrators to manage trusted context users in DB2. With EXTERNAL SECURITY PROFILE, profile-name was added to user clause to allow trusted context users to be managed by RACF.

When an unqualified name was used in your application, using the new syntax ROLE AS OBJECT OWNER AND QUALIFIER will cause the role value to be used as the default for CURRENT SCHEMA and CURRENT PATH special registers, as well as the default qualifier for the EXPLAIN tables.

**Note:** The message texts for the following SQLCODEs were changed due to APAR PK44617.

- ▶ -20373
- ▶ -20374
- ▶ -20422

Some value changes have been introduced as a result of this support:

- ▶ The initial value of the special registers CURRENT SCHEMA and CURRENT PATH for trusted connections with the ROLE AS OBJECT OWNER AND QUALIFIER clause in effect has changed:
  - CURRENT SCHEMA  
The initial value of the special register is the value of role name associated with the user in the trusted context, if the trusted connection is established with the ROLE AS OBJECT OWNER AND QUALIFIER clause in effect.
  - CURRENT PATH  
The initial value of the special register is "SYSIBM", "SYSFUN", "SYSPROC", "value of role name associated with the user in the trusted context" if the PATH bind option or SQL PATH option is not specified and if the trusted connection with the ROLE AS OBJECT OWNER AND QUALIFIER clause in effect.
- ▶ The default schema of EXPLAIN tables for trusted connections with the ROLE AS OBJECT OWNER AND QUALIFIER clause in effect has changed.
  - The default schema is the role associated with the process if the EXPLAIN statement is executed in a trusted connection with the ROLE AS OBJECT OWNER AND QUALIFIER clause in effect.

## 4.3 Encryption options

In addition to the basic encryption options described in "DRDA encrypted passwords" on page 138, you can choose several options for stronger security. In this section we show different ways to set and use strong encryption:

- ▶ DRDA encryption
- ▶ IP Security
- ▶ Secure Socket Layer (AT-TLS)
- ▶ DataPower

### 4.3.1 DRDA encryption

Traditionally, DB2 used Data Encryption Standards (DES, FIPS 46-3). By installing APAR PK56287, DRDA access to DB2 V8 for z/OS or later now supports Advanced Encryption Standard (AES) user ID and password encryption, when establishing a connection with a DB2 for z/OS with a tighter security.

Now DB2 for z/OS supports two levels of encryption. Even though DES weak cryptographic function, DB2 supports DES encryption/decryption for compatibility reasons with downlevel systems.

- ▶ DES  
56-bit single DES encryption of the password and the Diffie-Hellman algorithm to generate a key for the encryption algorithm at connect time.
- ▶ AES  
256-bit AES encryption and the Diffie-Hellman algorithm to generate a key for the encryption algorithm.

**Note:** DB2 for LUW Version 8, FixPack 16, the IBM Data Server Driver for JDBC and SQLJ provides support for AES encryption for connecting to DB2 for z/OS V8 or later with appropriate PTF.

AES encryption applies to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity. You request AES encryption by setting the IBM DB2 Driver for JDBC and SQLJ property encryptionAlgorithm.

DB2 Connect V9.1 FP5, V9.5 FP3 or later and equivalent level of data servers drivers provides AES encryption.

#### Using AES encryption to connect DB2 for z/OS

DB2 for z/OS requires to have ICSF on z/OS to use AES encryption. If ICSF is not enabled, you will see the message in Example 4-15.

*Example 4-15 SYSLOG output from misconfiguration*

---

```
DSNL046I  -D9C3 DSNLTSEC ICSF NOT ENABLED
```

---

**Note:** To enable AES encryption, you need to enable ICSF with 256-bit AES encryption. In the course of our installation, we found we needed a new level of microcode level for our z9® cryptographic coprocessor to add AES support and the z/OS V1R10 ICSF APAR OA27145 (PTF UA45350)

After you have configured the server to enable AES encryption, you choose the option to use AES encryption from the database directory, using the command shown in Example 4-16.

*Example 4-16 Catalog database with AES option*

---

```
$ db2 catalog db DB9C at node WTSC63 authentication SERVER_ENCRYPT_AES
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
```

---

If you are using one of the data server drivers to connect to DB2 for z/OS using AES encryption, you can pass your option using the *Authentication* option of the IBM Data Server Driver configuration keywords. Example 4-17 shows a configuration using AES encryption for the data server driver. You can also choose the option through connection attributes.

*Example 4-17 Sample data server driver configuration for AES encryption*

---

```
<configuration>
  <DSN_Collection>
    </dsn>
    <dsn alias="DB9A" name="DB9A" host="wtsc63.itso.ibm.com" port="12347">
      <parameter name="Authentication" value="Server_encrypt_aes"/>
    </dsn>
    <dsn alias="DB9C" name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="Authentication" value="Server_encrypt"/>
    </dsn>
  </DSN_Collection>
  ...
</configuration>
```

---

For Java applications, the encryptionAlgorithm driver property provides the option to choose between the 56-bit DES encryption (encryptionAlgorithm value of 1) and the 256-bit AES encryption (encryptionAlgorithm value of 2). Example 4-18 shows how to set the property from DB2DataSource class method.

**Important:** To use AES encryption in Java environment, you need to obtain the unrestricted policy file for JCE as documented in the manual. It is available at the following Web page:

<https://www.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk>

*Example 4-18 Using AES from Java application*

---

**Set following property to use AES(2)**

```
(DB2DataSource)db2ds.setEncryptionAlgorithm(2);
```

**one of the following two security options supports AES encryption**

```
db2ds.setSecurityMechanism(com.ibm.db2.jcc.DB2BaseDataSource.ENCRYPTED_PASSWORD_SECURITY);
db2ds.setSecurityMechanism(com.ibm.db2.jcc.DB2BaseDataSource.ENCRYPTED_USER_AND_PASSWORD_SECURITY);
```

---

If you are configuring WebSphere Application Server to use AES configuration, you can add the DataSource property from the DataSource custom properties panel of WebSphere Application Server admin console, as shown in Figure 4-9 on page 149.



<input type="checkbox"/>	<a href="#">setEncryptionAlgorithm</a>	<a href="#">2</a>	<a href="#">false</a>
--------------------------	--	-------------------	-----------------------

Figure 4-9 Configure WebSphere Application Server DataSource custom properties to use AES encryption

**Note:** Enable the IBM Java Cryptography Extension (JCE) on your client. The IBM JCE is part of the IBM SDK for Java, Version 1.4.2 or later. For earlier versions of the SDK for Java, you need to install the ibmjceprovider.jar package.

## Data stream encryption options

DB2 provides a DRDA data stream encryption option, however DRDA data encryption only supports 56-bit DES encryption to encrypt/decrypt data. Future direction for data stream encryption is to utilize SSL support by DB2 9 for z/OS. See 4.3.3, “Secure Socket Layer” on page 151, for detail on SSL connection. Using data stream encryption requires the Integrated Cryptographic Service Facility (ICSF) enablement on z/OS.

When using DB2 Connect, you specify the data stream encryption through database catalog command, as shown in Example 4-19.

Example 4-19 Catalog database with DRDA data stream encryption

```
$ db2 catalog db DB9A at node WTSC63 authentication DATA_ENCRYPT
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
```

Example 4-20 shows the setting for data stream encryption for an ODBC/CLI/.NET application with data server drivers.

Example 4-20 Sample data server driver configuration for data stream encryption

```
<configuration>
  <DSN_Collection>
    </dsn>
    <dsn alias="DB9A" name="DB9A" host="wtsc63.itso.ibm.com" port="12347">
      <parameter name="Authentication" value="Data_encrypt"/>
    </dsn>
    <dsn alias="DB9C" name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="Authentication" value="Server_encrypt"/>
    </dsn>
  </DSN_Collection>
  ...
</configuration>
```

For Java applications, the T4 driver provides two options for data stream encryption under the securityMechanism driver property. See Example 4-21 on page 150. You can set the data stream encryption from your Java application or WebSphere Application Server admin console.

**encryptionAlgorithm of 1(DES) is only supported(optional)**

```
(DB2DataSource)db2ds.setEncryptionAlgorithm(1);
```

**one of the following two security options supports data stream encryption**

```
db2ds.setSecurityMechanism(com.ibm.db2.jcc.DB2BaseDataSource.ENCIPHERED_USER_AND_DATA_SECURITY);
```

```
db2ds.setSecurityMechanism(com.ibm.db2.jcc.DB2BaseDataSource.ENCIPHERED_USER_PASSWORD_AND_DATA_SECURITY);
```

### 4.3.2 IP Security

IP Security (IPSec) is a set of protocols and standards defined by the Internet Engineering Task Force (IETF), which provides open architecture for security at the IP networking layer of TCP/IP. IPSec has been deployed widely to implement Virtual Private Network (VPN).

Because IPSec works on the IP networking layer, IPSec can be used to provide security for any TCP/IP applications without modifications, including DB2 for z/OS. If necessary, applications can have their own security mechanisms on top of IPSec. Figure 4-10 illustrates how IPSec works with DB2 for z/OS server and DRDA AR clients. The Transport Layer protocols of the Internet Protocol Suite such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) use ports as communications endpoints. A specific port is identified by its number, commonly known as the port number, the IP address it is associated with, and the protocol used for communication.

UDP provides a simple message service for transaction-oriented services. Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts. Internet Control Message Protocol (ICMP) messages contain information about routing with IP datagrams or simple exchanges such as time-stamp or echo transactions.

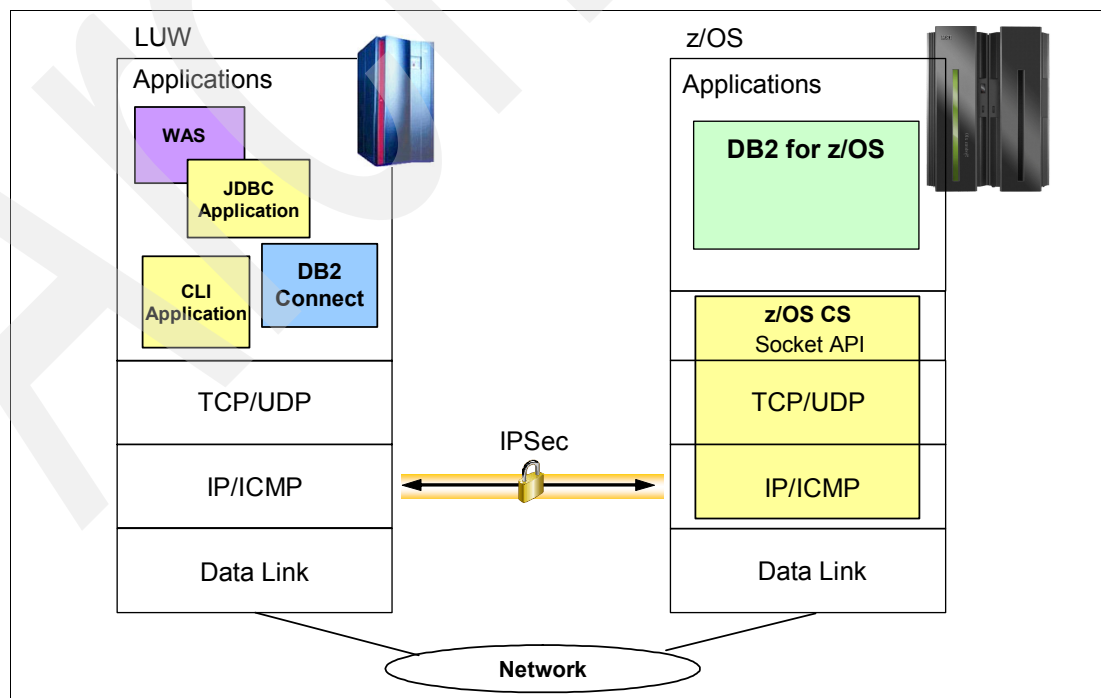


Figure 4-10 IPSec overview

IPSec works in the IP layer and network traffic will be encrypted. Data will be passed to the IP layer in clear text. Using IPSec does not require any changes to existing applications.

IPSec will utilize the System z cryptographic hardware, if the hardware is enabled and the required cryptographic algorithm is supported by the hardware. In addition, IPSec is zIIP eligible: the zIIP IPSECURITY (zIIP assist for IPSec) function can reduce IPSec processing load on General Processors well beyond what is achievable using cryptographic hardware.

There are no DB2 for z/OS settings related to IPSec. For details, refer to the following publications:

- *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-75350
- *IBM z/OS V1R10 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-76990

### 4.3.3 Secure Socket Layer

Secure Socket Layer (SSL), is a protocol designed and implemented by Netscape in response to growing concerns over security on the internet. SSL was first implemented to secure network traffic between browser and server. For example, when you are using Internet Banking, you see a little padlock symbol on bottom of your browser (see Figure 4-11), which tells you are using SSL. Other applications such as TELENT or FTP started using SSL and it has become a general solution for network security. SSL does encryption between two applications, where IPSec encrypt the whole network between hosts.

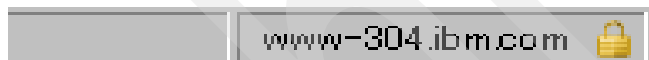


Figure 4-11 The padlock symbol indicates encryption

DB2 9 for z/OS supports SSL connections from DRDA AR clients also supporting SSL. SSL provides encryption of data on the wire. On System z, DB2 for z/OS uses the z/OS Communication Server (z/OS CS) IP Application Transparent Transport Layer service (AT-TLS). z/OS V1R7 CS IP introduced a new AT-TLS function in the TCP/IP stack to provide TLS for TCP/IP sockets applications that require secure connections. AT-TLS performs TLS on behalf of the application by invoking the z/OS System SSL in the TCP layer of the stack.

The z/OS CS Policy Agent is a started task that is used to manage the policies that are defined by the network administrators for their users. An AT-TLS policy is a file that defines the SSL characteristics of a connection that the AT-TLS can understand to invoke the z/OS system SSL. Figure 4-12 on page 152 illustrates how the SSL connection works between DB2 for z/OS server and DRDA AR clients.

z/OS System SSL provides support for SSL 2.0, SSL 3.0 and TLS 1.0.

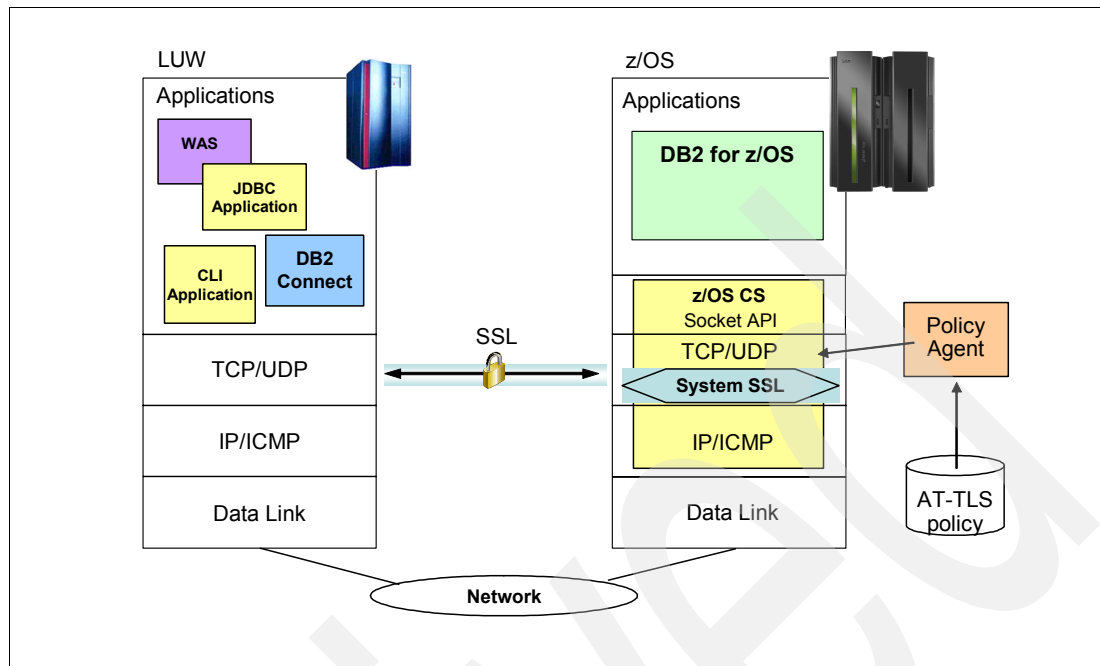


Figure 4-12 SSL overview

SSL consists of the record protocol and the handshake protocol. The record protocol controls the flow of the data between the two endpoints of an SSL session. The handshake protocol authenticates one or both ends of the SSL session and establishes a unique symmetric key used to generate keys to encrypt and decrypt data for that SSL session. SSL uses asymmetric cryptography, digital certificates, and SSL handshake flows to authenticate one or both end of the SSL session. A digital certificate can be assigned to the applications using SSL on each end of the connection. The digital certificate is comprised of a public key and some identifying information that has been digitally signed. Each public key has an associated private key, which is not stored with or as part of the certificate. The applications which is being authenticated must prove that it has access to the private key associated with the public key contained within the digital certificate.

There are three ways to prepare a certificate for use in SSL/TLS connections:

- ▶ Request a well-known Certification Authority (CA) to sign your certificate
- ▶ Generate a certificate yourself
- ▶ Create a self-signed CA certificate and function as local CA

The following example shows how to configure the DB2 9 for z/OS server system using the internal CA signed site certificate with RACDCERT command. For other options, see *IBM z/OS V1R10 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7699.

In the following sections we show you how to perform the following tasks:

- ▶ Prepare to use SSL at DB2 9 for z/OS server
- ▶ Preparing to use SSL for Java clients
- ▶ Preparing to use SSL for non-Java-based Clients

### Prepare to use SSL at DB2 9 for z/OS server

Preparing to use SSL includes defining policy agent and TCP/IP stack configuration. Related RACF resource definitions include access to policy agent and RACF keyrings and SSL certificates, and DB2 for z/OS configuration. The sample resource definitions are in no

specific order, you may change your order as needed. We will not go into details for each parameter. Refer to *z/OS V1R10.0 Communication Server: IP Configuration Reference*, SC31-8776.

Setup on z/OS includes the following resources:

- ▶ RACF setup
  - Define and permit RACF resource for Policy Agent
  - Define RACF keyrings and SSL certificates
- ▶ TCP/IP setup
  - PROFILE.TCPIP configuration
  - TCP/IP stack initialization access control
  - AT-TLS policy configuration
- ▶ DB2 for z/OS setup

*Figure 0-1 BSDS*

**Important:** We recommend applying the PTF corresponding to PK81175 when setting up the DB2 for z/OS server to use SSL connections

Figure 4-13 on page 154 shows the sets of RACF commands that define the RACF resources needed for SSL access and their access permits. Definitions include creation of a user for the policy agent, the syslog daemon, granting access to TCP/IP stack and related commands. The RACF resource EZB.INITSTACK.sysname.tcpname in the SERVAUTH class is used to block stack access, except for the user IDs permitted to access the resource.

**Define user ID assign to policy agent and syslog daemon**

```
ADDUSER PAGENT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
ADDUSER SYSLOGD DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
RDEFINE STARTED PAGENT.* STDATA(USER(PAGENT))
RDEFINE STARTED SYSLOGD.* STDATA(USER(SYSLOGD))
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED) REFRESH
```

**Restricting access to operator commands**

```
SETROPTS RACLIST (OPERCMDS)
RDEFINE OPERCMD (MVS.SERVGR.PAGENT) UACC(NONE)
PERMIT MVS.SERVGR.PAGENT CLASS(OPERCMDS) ACCESS(CONTROL) ID(PAGENT)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

**Use to block stack access except permitted user**

```
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE SERVAUTH EZB.INITSTACK.SC63.TCPIP UACC(NONE)
PERMIT EZB.INITSTACK.SC63.TCPIP -
CLASS(SERVAUTH) ID(OMVSKERN) ACCESS(READ)
PERMIT EZB.INITSTACK.SC63.TCPIP -
CLASS(SERVAUTH) ID(PAGENT) ACCESS(READ)
PERMIT EZB.INITSTACK.SC63.TCPIP -
CLASS(SERVAUTH) ID(SYSLOGD) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

**Allow access to the pasearch command**

```
RDEFINE SERVAUTH EZB.PAGENT.SC63.TCPIP.* UACC(NONE)
PERMIT EZB.PAGENT.SC63.TCPIP.* -
CLASS(SERVAUTH) ID(SYSADM) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Figure 4-13 Define RACF resources for policy agent

You need to create a started task for policy agent named PAGENT in your installation of PROCLIB library, Figure 4-14 on page 155 shows a sample definition for PAGENT procedure.

**Note:** In our test environment, the configuration files are defined as UNIX System Services files. Alternatively, you may choose to prepare your configuration files as MVS data sets. If that is the case, STDENV DD can be defined as below. The same applies to the other configuration files.

```
//STDENV DD DSN=SYS1.TCPPARMS(PAENV),DISP=SHR
```

```
//PAGENT PROC
//*
//* SecureWay Communications Server IP
//* SMP/E distribution name: EZAPAGSP
//*
//* 5647-A01 (C) Copyright IBM Corp. 1999.
//* Licensed Materials - Property of IBM
//*
//STEP0 EXEC PGM=BPXTCAFF,PARM=TCPIP
//PAGENT EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV DD PATH='/u/stc/tls/pagent.sc63.soap.env',
//          PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 4-14 Definition for policy agent started task

You define policy agent environment file specified in your started task, as shown in Figure 4-15. A brief explanation for each specified parameter follows:

- ▶ **PAGENT\_CONFIG\_FILE:** Point to a policy agent main configuration file.
- ▶ **PAGENT\_LOG\_FILE:** Specifies the destination of the log file.
- ▶ **TZ:** Set to local time, our installation point to Eastern time zone in the United States.

```
Policy agent environment file, /u/stc/tls/pagent.sc63.soap.env
PAGENT_CONFIG_FILE=/u/stc/tls/pagent.sc63.soap.conf
PAGENT_LOG_FILE=/tmp/pagent.sc63.soap.log
TZ=EST5EDT
```

Figure 4-15 Definition for policy agent environment file

Define the policy agent main configuration file, which points to AT-TLS configuration file, as shown in Figure 4-16. A brief explanation for each specified parameter follows:

- ▶ **LogLevel**  
Specify the level of tracing for the Policy Agent. For example, if you specify 15, syserr, objerr, proterr, and warning will be written.
- ▶ **TcpImage**  
Specify a TCP/IP image and its associated image configuration file to be installed to that image.
- ▶ **TTLScnfig**  
Specify the path of a local AT-TLS policy file that contains stack-specific AT-TLS policy statements

```
Policy agent main configuration file, /u/stc/tls/pagent.sc63.soap.conf
LogLevel 15
TcpImage TCPIP flush purge 1
TTLScnfig /u/stc/tls/pagent.sc63.soap.policy flush purge
```

Figure 4-16 Definition for policy agent main configuration file

Define DB2 9 for z/OS as a SSL server in the AT-TLS configuration file as shown in Figure 4-17. In our installation, the security port is 12349 and the IP address used to access specific server is 9.12.6.70. The keyring name is case sensitive.

► **TTLRule**

Define an AT-TLS rule. Specify DB2 for z/OS secure port number to LocalPortRange, and IP address used to access DB2 for z/OS to LocalAddr. Jobname is name of application which will be DDF address space, ssidDIST.

► **TTLGroupAction**

Specify parameters for a Language Environment process required to support secure connections. 'TTLEnabled On' means AT-TLS security is active, which Data might be encrypted, based on other policy statements.

► **TTLEnvironmentAction**

Specify the attributes for an AT-TLS environment. HandshakeRole specifies the SSL handshake role to be taken.

```
configuration for AT-TLS configuration file, /u/stc/tls/pagent.sc63.soap.policy
## -----
# Server Rule DB9ADIST on SC63, SSL-port 12349, 9.12.6.70
## -----
TTLRule DB9ADIST_12349
{
    LocalPortRange 12349
    LocalAddr 9.12.6.70
    Jobname DB9ADIST
    Userid STC
    Direction INBOUND
    TTLGroupActionRef DB9ADISTGrpAct
    TTLEnvironmentActionRef DB9ADISTEnvAct
}
## -----
# DB9ADIST Server Group action
## -----
TTLGroupAction DB9ADISTGrpAct
{
    TTLEnabled On
    Trace 15
}
## -----
# DB9ADIST Server Environment action
## -----
TTLEnvironmentAction DB9ADISTEnvAct
{
    TTLSKeyRingParms
    {
        Keyring DB9AKEYRING
    }
    HandShakeRole Server
}
```

Figure 4-17 Sample server definition for AT-TLS configuration file



Modify TCPIP.PROFILE with TPCCONFIG TTLS as shown in Figure 4-18. When the TTLS subparameter is specified, the procedure starts after the policy agent has successfully installed the AT-TLS policy in the TCP/IP stack and AT-TLS services are available.

**TCP/IP configuration file, TCPIP.PROFILE**

```
TCPCONFIG      RESTRICTLOWPORTS
                TTLS
```

*Figure 4-18 Add TTLS parameter to TCP/IP stack configuration*

Before you create keyrings and certificates, activate the DIGTCERT and DIGTRING generic classes. Define and permit access to resources using the command in Figure 4-19.

```
SETR CLASSACT(DIGTCERT DIGTRING)
RDEF FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
RDEF FACILITY IRR.DIGTCERT.LIST      UACC(NONE)
PE IRR.DIGTCERT.LIST      CLASS(FACILITY) ID(SYSDSP) ACCESS(CONTROL)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(SYSDSP) ACCESS(READ)
SETR RACLIST (DIGTRING DIGTCERT) REFRESH
```

*Figure 4-19 Activate DIGTCERT and DIGTRING class*

Create a self-signed server CA certificate, using the RACDCERT RACF command. Figure 4-20 shows the creation of a sample self-signed certificate called DB9ASSLCA.

```
RACDCERT CERTAUTH GENCERT +
  SUBJECTSDN( +
    CN('wtsc63.itso.ibm.com') +
    OU('UTEC620') +
    O('IBM') +
    L('SVL') +
    SP('CA') +
    C('USA') ) +
  SIZE(1024) +
  NOTBEFORE(DATE(2008-08-14)) +
  NOTAFTER(DATE(2030-12-31)) +
  WITHLABEL('DB9ASSLCA') +
  KEYUSAGE(CERTSIGN) +
  ALTNAME(DOMAIN('wtsc63.itso.ibm.com')) )
```

*Figure 4-20 Create a self-signed server CA certificate*

Similarly, create a private server certificate using the RACDCERT RACF command. Figure 4-21 on page 158 shows the sample private server certificate called DB9ASSLCERT for user ID STC.

```

RACDCERT ID(STC) GENCERT +
  SUBJECTSDN( +
    CN('wtsc63.itso.ibm.com') +
    OU('UTEC620') +
    O('IBM') +
    L('SVL') +
    SP('CA') +
    C('USA') ) +
  SIZE(1024) +
  NOTBEFORE(DATE(2008-08-14)) +
  NOTAFTER(DATE(2030-12-31)) +
  WITHLABEL('DB9ASSLCERT') +
  ALTNAME(DOMAIN('wtsc63.itso.ibm.com')) ) +
  SIGNWITH(CERTAUTH LABEL('DB9ASSLCA'))

```

Figure 4-21 Create private server certificate

Create a server keyring and add the created certificate, Figure 4-22 shows the creation of keyring DB9AKEYRING and addition of DB9ASSLCA and DB9ASSLCERT to the keyring.

**create a keyring and add certificates to created keyring.**

```

RACDCERT ID(STC) ADDRING(DB9AKEYRING)
RACDCERT ID(STC) +
  CONNECT(CERTAUTH LABEL('DB9ASSLCA') +
  RING(DB9AKEYRING) )
RACDCERT ID(STC) +
  CONNECT(LABEL('DB9ASSLCERT') +
  RING(DB9AKEYRING) +
  DEFAULT)

```

**To verify your definition, issue following command, follows by sample output.**

```
RACDCERT ID(STC) LISTRING(DB9AKEYRING)
```

Digital ring information for user STC:

Ring:

>DB9AKEYRING<

Certificate Label Name	Cert Owner	USAGE	DEFAULT
DB9ASSLCA	CERTAUTH	CERTAUTH	NO
DB9ASSLCERT	ID(STC)	PERSONAL	YES

Figure 4-22 Create server keyring and add certificate

After creating and adding keyring completes, export the server CA certificate to a data set using the command shown in Figure 4-23 on page 159. The exported certificate will be given to the DRDA AR clients.

```
RACDCERT CERTAUTH +  
  EXPORT(LABEL('DB9ASSLCA')) +  
  DSN('PAOLOR7.DB9ASSLC.B64') +  
  FORMAT(CERTB64)
```

Figure 4-23 Export server CA certificate

Set up your DB2 9 for z/OS to open the secure port. Update your BSDS using the Change Log Inventory utility (DSNJU003). Figure 4-24 show the setting of secure port 12349.

```
//DSNJU003 JOB ('DSNJU003'),'CHANGE LOG INV',CLASS=A,MSGLEVEL=(1,1)  
//STEP010 EXEC PGM=DSNJU003  
//STEPLIB DD DSN=DB9A9.SDSNEXIT,DISP=SHR  
//          DD DSN=DB9A9.SDSNLOAD,DISP=SHR  
//SYSUT1 DD DSN=DB9AU.BSDS01,DISP=OLD  
//SYSUT2 DD DSN=DB9AU.BSDS02,DISP=OLD  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
DDF LOCATION=DB9A,SECPORT=12349  
/*
```

Figure 4-24 Change BSDS to enable the secured port

**Note:** Consider the following rules when configuring SECPORT to DB2 for z/OS:

- ▶ If the value of SECPORT is disabled, the client can still use the DRDA PORT and use SSL on it, but DB2 for z/OS does not validate whether the connection uses SSL protocol.
- ▶ DB2 9 for z/OS will not allow SSL to work with the TCP/IP BIND specific statements. Use DDF IPV4 or IPV6 and GRPIPV4 or GRPIPV6 statements instead to use a specific IP address with DB2 for z/OS.

For details of each command and parameter, refer to following documents:

- ▶ *z/OS V1R10 Communications Server IP: Configuration Guide*, SC31-8775-14
- ▶ *z/OS V1R10 Communications Server IP: Configuration Reference*, SC31-8776-15
- ▶ *z/OS V1R10 Communications Server IP: System Administrator's Commands*, SC31-8781-08
- ▶ *z/OS V1R10 Security Server RACF Command Language Reference*, SA22-7687-12
- ▶ *DB2 Version 9.1 for z/OS Utility Guide and Reference*, SC18-9855

## Preparing to use SSL for Java clients

To use SSL connection from your Java clients, you need to have the DB2 server certificate imported using the keytool command.

The sequence for creating and importing your DB2 server certificate is follows:

1. Download the DB2 server certificate from the server.

RACF creates DB2 server certificate in EBCDIC. Be sure to FTP the DB2 server certificate in ASCII mode from the Server. In our example, we named the downloaded certificate “cert.arm” in “C:\temp\residency” directory.

2. (Optional) Create the keystore using keytool command.

To create new keystore, use the -genkey option. In our case, see Example 4-22, the name “keystore” is used for the keystore.

**Note:** Example 4-22 omits information for keystore and just shows the steps. It is recommended to give appropriate information when creating your keystore.

*Example 4-22 Generating keystore for Java clients*

---

**Create a keystore for your application(or use your environment keystore)**

```
$ ./keytool -genkey -keystore keystore
Enter keystore password: passwd123
What is your first and last name?
[Unknown]: PAOLOR7
What is the name of your organizational unit?
[Unknown]: ITS0
What is the name of your organization?
[Unknown]: IBM
What is the name of your City or Locality?
[Unknown]: SVL
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=PAOLOR7, OU=ITS0, O=IBM, L=SVL, ST=CA, C=US correct? (type "yes" or "no")
[no]: yes

Enter key password for <mykey>:
(RETURN if same as keystore password):
```

---

3. Import the DB2 server certificate to a keystore.

Import the DB2 server certificate using the -import option. See Example 4-23 on page 161.

We use also the following options to import the DB2 server certificate:

- -alias option

This option provides label name for the DB2 server certificate.

- -file option

This option provides name of the DB2 server certificate download in 1., “Download the DB2 server certificate from the server.” on page 159. In the example “cert.arm” is used.

- -keystore option

This option provides name or full path to the keystore. In the example, “keystore” is used as the keystore name and keytool command was executed where the keystore resides.

You will be prompted to enter the keystore password.

*Example 4-23 Import server certificate to your keystore*

---

**Importing certificate FTPed from z/OS to keystore**

```
$ keytool -import -alias DB9ASSLCA -file cert.arm -keystore keystore
Enter keystore password: passwd123
Owner: OU=DB9ASSL, O=IBM, L=SVL, ST=CA, C=USA
Issuer: OU=DB9ASSL, O=IBM, L=SVL, ST=CA, C=USA
Serial number: 0
Valid from: 8/13/08 11:00 PM until: 12/31/30 10:59 PM
Certificate fingerprints:
    MD5: 93:AF:C4:9B:96:E4:B2:A0:9C:7F:B3:EC:8E:6C:CD:8A
    SHA1: D5:15:65:E0:A8:49:2C:46:7A:50:00:38:1A:B9:05:1A:9E:F3:1A:33
Trust this certificate? [no]: yes
Certificate was added to keystore
```

---

You can now use the -list option to verify the importing of the DB2 server certificate to the keystore. Example 4-24 shows the output from keytool command with “db2assaca” entry.

*Example 4-24 List the keystore entry*

---

```
$ keytool -keystore keystore -list
Enter keystore password: passwd123

Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

db9asslca, Apr 8, 2009, trustedCertEntry,
Certificate fingerprint (MD5): 93:AF:C4:9B:96:E4:B2:A0:9C:7F:B3:EC:8E:6C:CD:8A
mykey, Apr 8, 2009, keyEntry,
Certificate fingerprint (MD5): 15:D4:87:3F:1B:AC:EF:03:AA:67:E0:07:34:4E:69:8B
```

---

## **Preparing to use SSL for non-Java-based Clients**

To use an SSL connection from non-Java-based clients, you need to import the DB2 server certificate using IBM Key Management tool, which is part of DB2 Global Security Kit (GSKit). If the GSKit is not available on the system, instructions are given in *Configuring Secure Sockets Layer (SSL) support in the DB2 client*, available from the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/t0053518.htm>

Create and import your DB2 server certificate as follows:

1. Download the DB2 server certificate from the server.

RACF creates DB2 server certificate in EBCDIC. Be sure to FTP the DB2 server certificate in ASCII mode from the Server. In our example, we name it “cert.arm” in “C:\temp\residency” directory.

2. Start the IBM Key Management tool.

GSKit requires you to set the JAVA\_HOME environment variable before you start the IBM Key Management tool. The results if you start your IBM Key Management tool without setting the JAVA\_HOME, is shown in Example 4-25 on page 162.

*Example 4-25 Starting the IBM Key Management tool*

```
$ export JAVA_HOME=/usr/java5
$ gsk7ikm_64
```

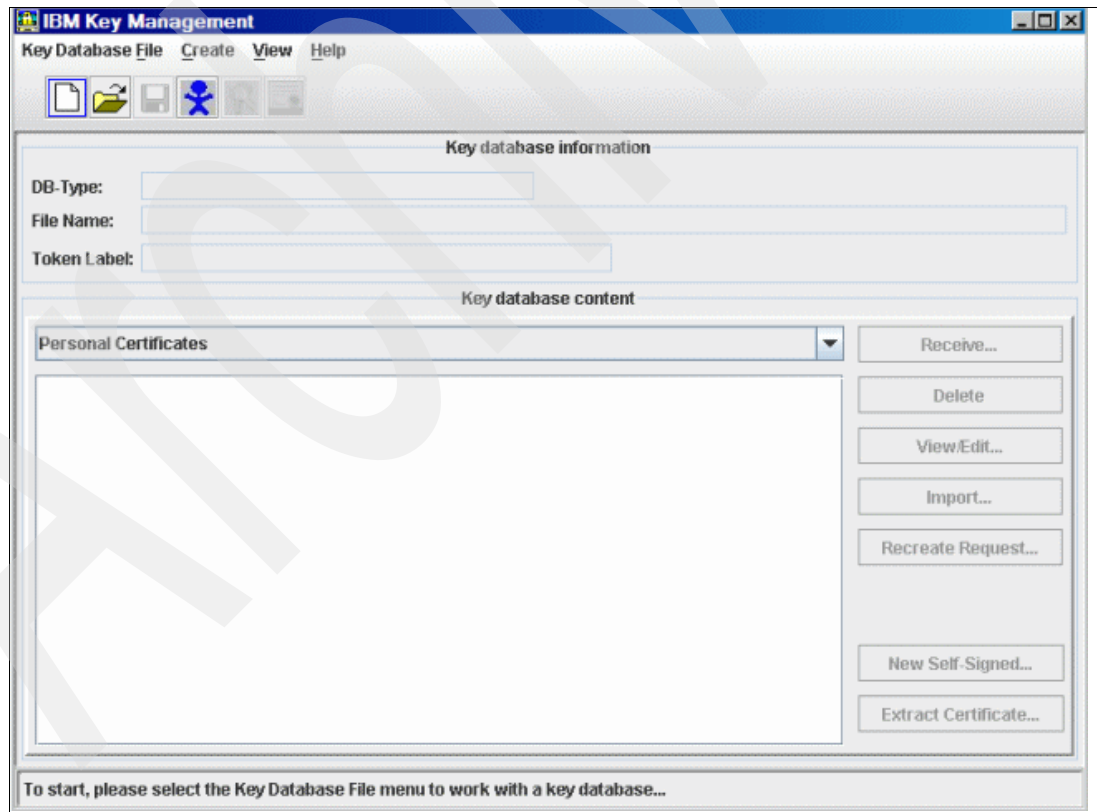
**Probable error message without setting JAVA\_HOME environment variable**

```
$ gsk7ikm_64
which: 0652-141 There is no java in /usr/bin /etc /usr/sbin /usr/ucb
/home/db2inst1/bin /usr/bin/X11 /sbin . /home/db2inst1/sqlllib/bin
/home/db2inst1/sqlllib/adm /home/db2inst1/sqlllib/misc.
java is not found
please set JAVA_HOME or add java to path
For example, you can set JAVA_HOME in a korn shell by the
following command
    export JAVA_HOME=/usr/jdk1.4
where JDK is installed in directory /usr/jdk1.4
```

**Tip:** In Windows platform, you can set your JAVA\_HOME environment variable using following command:

```
set JAVA_HOME=C:\Program Files\IBM\Java50
```

After issuing the command, the IBM Key Management tool window appears. See Figure 4-25.



*Figure 4-25 The IBM Key Management tool window*

3. (Optional) Create new Key Database file.

The Key database type field should be “CMS”. To create a new Key Database file, click **Key Data File**, then click **New** in the IBM Key Management tool. The dialog shown in Figure 4-26 will appear where you select and input three fields.

- Key database type: You must select “CMS”.
- File name: Specify the name of the keystore, In the example “key.kdb” is used.
- Location: Specify the directory of the keystore, In the example, C:\Program File\IBM\gsk7\bin\test\ is used.

**Note:** Be sure to place the keystore in a secure place and that the clients are accessible to the keystore.



Figure 4-26 Create a new Key Database file

Click **OK**. The “Password Prompt” dialog box (Figure 4-27 on page 164) will display. Provide the password for the keystore.

Select the **Stash the password to a file?** check box when you set the keystore password. This will create the stash file with the keystore. The stash file stores the password to access the keystore file in encrypted format, giving an extra level of security in client environment.

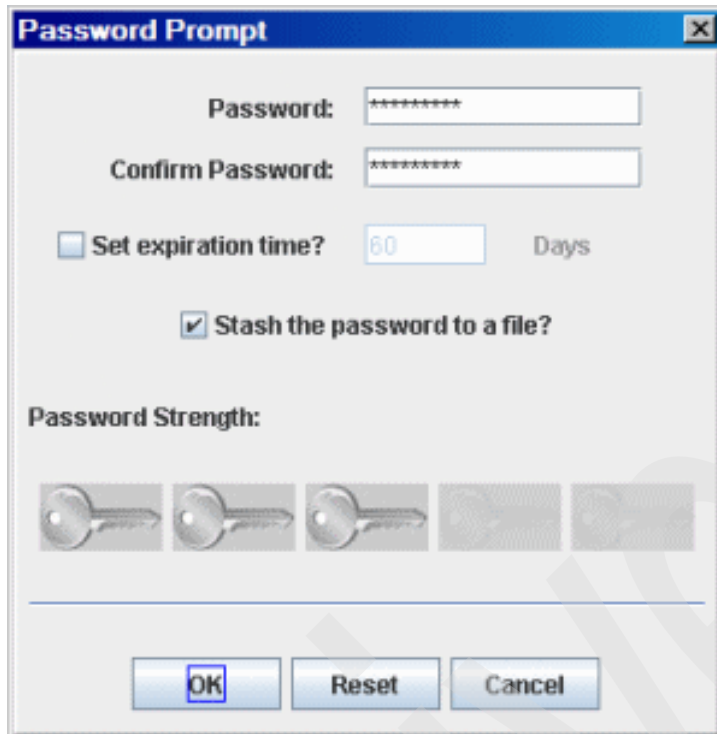


Figure 4-27 Setting password for key database and making stash file

4. Import the DB2 server certificate to a keystore.

After creating or opening your keystore into the IBM Key Management tool, click the **Add** button (Figure 4-28). Select the DB2 server certificate.

- Data type: Select “Base64-encoded ASCII data”.
- Certificate file name: Specify the name of the DB2 server certificate, In our example is “cert.arm” downloaded in 1., “Download the DB2 server certificate from the server.” on page 161.
- Location: Specify the directory of the DB2 server certificate.

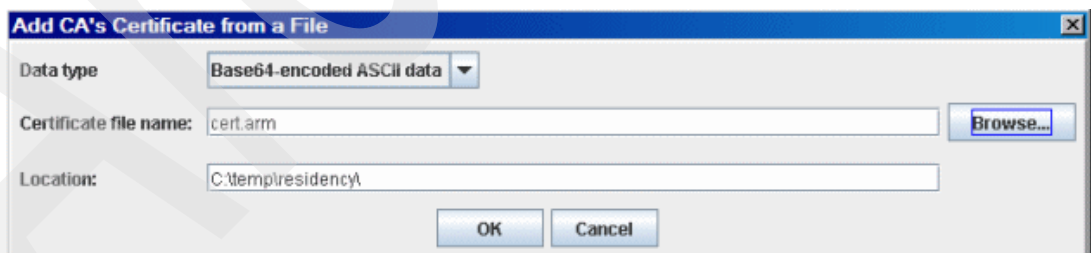


Figure 4-28 Import the DB2 server certificate to Key Database



Click **OK**. The “Enter a Label” dialog box (Figure 4-29) displays. Type in a label for the certificate to complete the import. In our example, “DB9ASSLCA” is used.



Figure 4-29 Enter the label for the DB2 server Certificate

After completing the import of the DB2 server certificate, your keystore should look like Figure 4-30, where the imported certificate is shown in Key database content. The keystore is now ready for use by your DB2 CLI applications.

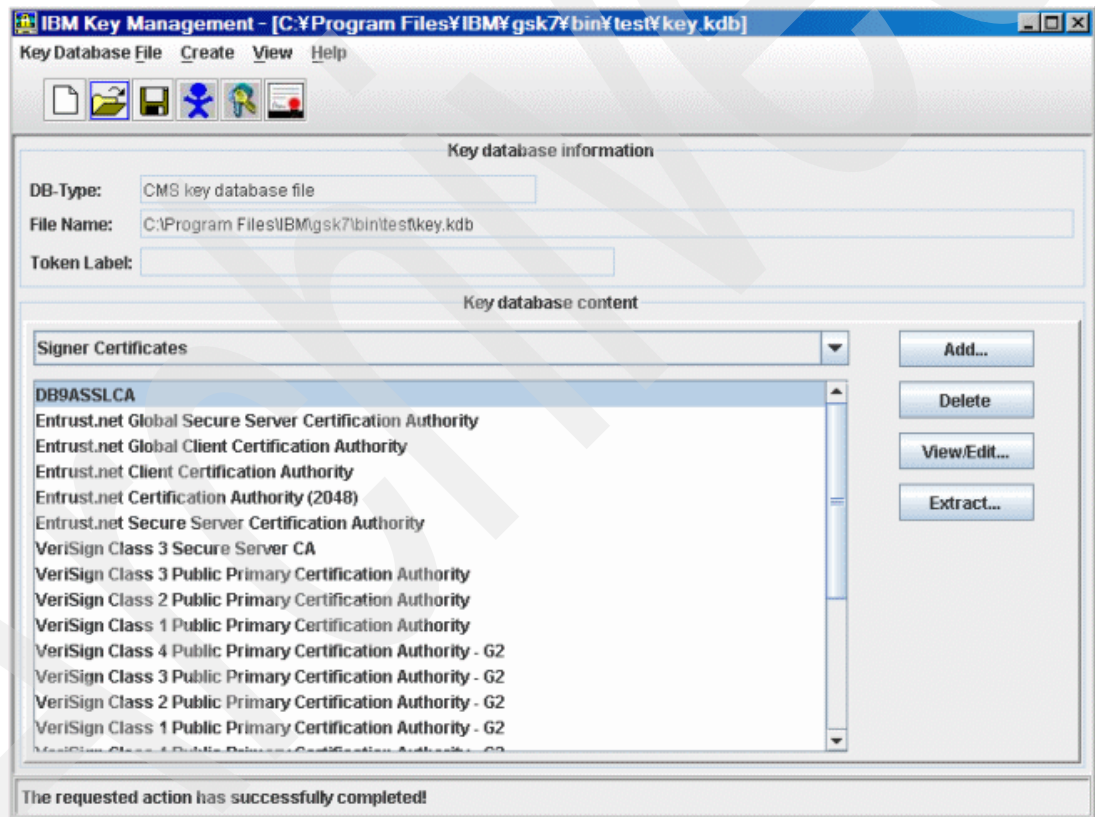


Figure 4-30 The key database after imported the DB2 server certificate

**Tip:** Alternatively, you can create the keystore using the **gsk7cmd** command from your command line. In the example below, name “key.kdb”, password “pwd123k”, type “CMS” and -stash option were specified.

```
$ gsk7cmd -keydb -create -db key.kdb -pw pwd123k -type cms -stash
$ ls
key.crl  key.kdb  key.rdb  key.sth
```

5. Create the SSL configuration file.

After importing the DB2 server certificates, you need to create the SSL configuration file in order for the DB2 CLI applications to use them. You should create the SSL configuration file named “SSLClientconfig.ini” in “\$INSTHOME/cfg”. In our example, the SSL configuration file is placed in ‘/home/db2inst1/sqllib/cfg’. The contents of the SSL configuration file are listed in Example 4-26.

– DB2\_SSL\_KEYSTORE\_FILE

Fully qualified file name of the KeyStore that stores the Server Certificate.

– DB2\_SSL\_KEYRING\_STASH\_FILE

Fully qualified file name to the stash file that stores the password to access the keystore file in encrypted format. This provides an extra level of security in the client scenario.

*Example 4-26 Contents of SSLClientconfig.ini*

---

```
DB2_SSL_KEYSTORE_FILE=/home/db2inst1/key.kdb
DB2_SSL_KEYRING_STASH_FILE=/home/db2inst1/key.sth
```

---

You are required to put SSL configuration file in a certain directory depending on the platform you are using.

UNIX: \$INSTHOME/cfg

Windows: \$INSTHOME/

An example of \$INSTHOME for Windows environment follows:

\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\DB2

The settings for \$DB2ISNTPORF and \$DB2INSTDEF DB2 profile variables are as shown in Figure 4-31.

```
C:\Program Files\IBM\SQLLIB\BIN>db2set -all
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[i] DB2INSTPROF=C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY1
[i] DB2COMM=TCPIP
[g] DB2_EXTSECURITY=NO
[g] DB2SYSTEM=LENOVO-B6AFDE0A
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS02
```

*Figure 4-31 Settings for \$DB2ISNTPORF and \$DB2INSTDEF DB2 profile variables*

**Note:** In DB2 Connect V9.7, you no longer need to use separate configuration file, SSLClientconfig.ini, to setup SSL support. Specification for the keystore file and stash file will be replaced by connection string keyword.

CLI/ODBC driver:

- ▶ ssl\_client\_keystoredb: Specify the fully-qualified key database file name.
- ▶ sl\_client\_keystash: Specify the fully-qualified stash file name.

DB2 .NET Data Provider:

- ▶ SSLClientKeystoredb: Specify the fully-qualified key database file name.
- ▶ SSLClientKeystash: Specify the fully-qualified stash file name.
- ▶ Security - Set security to SSL.

CLP and embedded SQL clients:

- ▶ The CATALOG TCPIP NODE command with SECURITY SSL parameter.
- ▶ The client-side database manager configuration parameters ssl\_clnt\_keydb, and ssl\_clnt\_stash to connect to a database using SSL.

## Using SSL from Java Application

To use SSL Connection from Java application, set the sslConnection property from Common IBM Data Server Driver for JDBC and SQLJ properties to "true", and connect to the security port as shown in Example 4-27.

*Example 4-27 sample Java code for SSL connection*

---

```
public static void main (String[] args)
{
    String ServerName = "wtsc63.itso.ibm.com";
    int PortNumber = 12349;
    String DatabaseName = "DB9A";
    java.util.Properties properties = new java.util.Properties();
    properties.put("user", userid);
    properties.put("password", mypasswd);
    properties.put("sslConnection", "true");
    String url = "jdbc:db2://" + ServerName + ":" + PortNumber + "/" + DatabaseName
    java.sql.Connection con = null;
    try
    {
        Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
    }
    catch ( Exception e )
    {
        System.out.println("Error: failed to load Db2 jcc driver.");
    }
    try
    {
        System.out.println("url: " + url);
        con = java.sql.DriverManager.getConnection(url, properties);
    }
}
```

application logic follows...

---

For Java applications, you can either pass the keystore name through JVM™ properties or Common IBM Data Server Driver for JDBC and SQLJ properties. Example 4-28 shows a way to pass the keystore name “keystore” and password “passwd123” as JVM properties. In “Using SSL from WebSphere Application Server” on page 171, we show how to pass that information using the Common IBM Data Server Driver for JDBC and SQLJ properties as a datasource settings.

*Example 4-28 Executing Java application using SSL*

```
$ java -Djavax.net.ssl.trustStore=keystore  
-Djavax.net.ssl.trustStorePassword=passwd123 SSLTest
```

## Using SSL from non-Java-based Applications using IBM DS Driver

If you are running your .NET applications on the application server, you can choose to install IBM Data Server Driver package and connect to DB2 for z/OS. The thin client package does not have a directory, so you need to configure a configuration file for the non-Java-based IBM Data Server Driver. Example 4-29 gives a sample configuration from our test environment. The dsn alias DB9AS is connection using SSL, which connect to Secure Port of our DB2, and the dsn alias DB9A shows settings for regular DRDA connection.

*Example 4-29 Sample db2dsdriver.cfg configuration*

```
<configuration>  
  <DSN_Collection>  
    <dsn alias="DB9AS" name="DB9A" host="wtsc63.itso.ibm.com" port="12349">  
      <parameter name="Authentication" value="Server_encrypt_aes"/>  
      <parameter name="SecurityTransportMode" value="SSL"/>  
    </dsn>  
    <dsn alias="DB9A" name="DB9A" host="wtsc63.itso.ibm.com" port="12347">  
      <parameter name="Authentication" value="Server_encrypt"/>  
    </dsn>  
  <databases>  
    <database name="DB9A" host="wtsc63.itso.ibm.com" port="12349">  
      <parameter name="ProgramName" value="PID"/>  
      <parameter name="ConnectionLevelLoadBalancing" value="true"/>  
      <parameter name="ClientUserID" value="clientuserid"/>  
      <parameter name="ClientApplicationName" value="my application name"/>  
      <parameter name="ClientWorkstationName" value="my workstation name"/>  
    </database>  
  </databases>  
  <parameters>  
    <parameter name="CommProtocol" value="TCPIP"/>  
  </parameters>  
</configuration>
```

**Note:** Our example configures AES encryption for user ID and password on top of SSL, which is a valid configuration. But there are no point in setting encryption over SSL.

## Using SSL from DB2 Connect client

In our example, DB2 Connect was used to connect to DB2 for z/OS using SSL. You will need to catalog your node directory specifying for the server the SECPORT of DB2 for z/OS and “security SSL”, as shown in Example 4-30 on page 169. Catalog database directory and DCS database directory the same way as your regular connection. All clients who use DB2 Connect to execute applications need to follow the same procedure to get the applications to use the SSL connection.

#### Example 4-30 Catalog DB2 server to use SSL connection to DB2 Connect

```
$ db2 catalog tcpip node sslnode remote wtsc63.itso.ibm.com server 12349 security SSL
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
$ db2 catalog db db9as at node sslnode authentication server_encrypt
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
$ db2 catalog dcs db db9as as db9a
DB20000I The CATALOG DCS DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
```

After completing the catalog of directories, verify your configuration by connecting to DB2 using the DB2 Connect Command Line Processor. You will not see any difference from your non-SSL connections, but verify your connection to the secure port of DB2 for z/OS. Our example shows the output from NETSTAT on z/OS. You should see that the connection was established to the DB2 security port, as shown in Example 4-31.

#### Example 4-31 Sample SSL connection using DB2 Connect

```
$ db2 connect to db9as user paolor7
Enter current password for paolor7:
```

##### Database Connection Information

```
Database server      = DB2 z/OS 9.1.5
SQL authorization ID = PAOLOR7
Local database alias = DB9AS
```

##### Netstat output shows Connection was established to DB2 Security Port

```
-DB9A DIS DDF
DSNL080I -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 278
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I TCPPORT=12347 SECPORT=12349 RESPORT=12348 IPNAME=-NONE
DSNL085I IPADDR=::9.12.6.70
DSNL086I SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL086I RESYNC  DOMAIN=wtsc63.itso.ibm.com
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

```
D TCPIP,,N,CONN
EZZ2500I NETSTAT CS V1R10 TCPIP 291
USER ID  CONN      LOCAL SOCKET          FOREIGN SOCKET        STATE
DB9ADIST 000037A6 0.0.0.0..12349        0.0.0.0..0            LISTEN
DB9ADIST 000037A5 0.0.0.0..12347        0.0.0.0..0            LISTEN
DB9ADIST 00005D30 9.12.6.70..12347      9.12.5.149..63362     ESTBLSH
DB9ADIST 00003970 9.12.6.70..12347      9.12.6.70..1072       ESTBLSH
DB9ADIST 00005D42 9.12.6.70..12347      9.12.4.121..44633     ESTBLSH
DB9ADIST 00004D38 9.12.6.70..12347      9.12.6.70..1073       ESTBLSH
DB9ADIST 000037E0 9.12.6.70..12347      9.30.28.163..4450     ESTBLSH
DB9ADIST 000037AF 0.0.0.0..12348        0.0.0.0..0            LISTEN
DB9ADIST 00005DAC 9.12.6.70..12349      9.12.5.149..63597     ESTBLSH
```

If you had connected to your DB2 secure port without properly setting all your configurations, you would have received an error message (as shown in Example 4-32 on page 170). In this case we have a connection failure because a non-SSL request was attempted to the DB2 security port.

Example 4-32 Sample SYSLOG output for failed SSL connection.

```
Connecting to DB2 Security port without "security SSL" option in node directory
$ db2 connect to db9as3 user paolor7
Enter current password for paolor7:
SQL30081N A communication error has been detected. Communication protocol being used:
"TCP/IP". Communication API being used: "SOCKETS". Location where the error was detected:
"9.12.6.70". Communication function detecting the error: "recv". Protocol specific error
code(s): "73", "*", "0".
SQLSTATE=08001
```

**Sample output from SYSLOG, which tells connection failure**

```
EZD1287I TTLS Error RC: 5003 Data Decryption 307
JOBNAME: DB9ADIST RULE: DB9ADIST_12349
USERID: STC GRPID: 0000000C ENVID: 00000000 CONNID: 00008187
DSNL511I -DB9A DSNLIENO TCP/IP CONVERSATION FAILED 308
TO LOCATION ::9.12.5.149
IPADDR::9.12.5.149 PORT=48584
SOCKET=RECV RETURN CODE=1121 REASON CODE=77B17343
```

If you use a network analyzer tool to capture the packets of the DRDA applications, you will clearly see the difference between having and not having the SSL encryption active. Figure 4-32 shows the network capture of a non-encrypted application. You can see that all the DRDA commands have been analyzed and data are shown as you execute the application. Anyone accessing the network with minimal skill can capture the data flowing across.

No.	Time	Source	Destination	Protocol	Info
1522	63.106316	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	38320 > 1504
1531	63.195889	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	EXCSATRD   A
1538	63.283815	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	SECCHKRM   A
1562	63.651737	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	SQLCARD
1563	63.651785	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
1566	63.652069	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
1567	63.652083	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	SQLDARD
1568	63.652092	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	QRYDSC
1569	63.652100	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	QRYDTA
1843	92.783824	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	ENDUOWRM   s
1846	92.864005	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	38320 > tsb2
1941	105.900240	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	ICMP	Echo (ping)
1951	106.900592	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	ICMP	Echo (ping)
1966	107.905197	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	ICMP	Echo (ping)
1974	108.904634	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	ICMP	Echo (ping)

Figure 4-32 The network capture of DRDA request (using Wireshark)

Figure 4-33 on page 171 shows an example of network capture of application with DRDA data encryption. The DRDA data encryption is done within DRDA, so the analyzer tool can still show the DRDA commands, but you will not be able to see data within DRDA command.

Intel(R) 82566MM Gigabit Network Connection (Microsoft's Packet Scheduler) : Capturing - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: `ip.src == 9.126.70` Expression... Clear Apply

No. ->	Time	Source	Destination	Protocol	Info
12115	685.181074	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	Unknown (0xb
12116	685.181688	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	QRNDSC
12119	685.182005	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12120	685.262022	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12121	685.262069	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12124	685.262324	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12125	685.262338	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12126	685.262348	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12129	685.262653	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12130	685.262667	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12133	685.342397	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12134	685.342812	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12137	685.343147	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12138	685.343157	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	TCP	[TCP segment
12139	685.343167	wtsc63.itso.ibm.com	LENOVO-B6AFDE0A-00903	DRDA	Unknown (0xb

Figure 4-33 The network capture of DRDA Data encrypt

Figure 4-34 shows an example of network capture of DRDA application with SSL enabled. As you can see, everything under TCP/IP is encrypted, so you are not able to see any DRDA commands with the network analyzer tool.

capture.txt - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: `ipdst == 9.126.70` Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
71	12.479706	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TCP	tabula > 123
72	12.479737	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TCP	[TCP Dup ACK
73	12.482030	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Application
74	12.482058	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	[TCP out-of-
76	12.683532	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Application
77	12.683581	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	[TCP out-of-
81	12.889579	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TCP	tabula > 123
82	12.889626	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TCP	[TCP Dup ACK
84	12.991286	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Application
85	12.991338	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	[TCP out-of-
87	13.260438	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Application
88	13.260486	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	[TCP out-of-
91	13.454375	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Application
92	13.454417	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	[TCP out-of-
95	13.647114	AA2024162.dhcp.makuha	wtsc63.itso.ibm.com	TLSv1	Encrypted A

Figure 4-34 The network capture of DRDA with SSL enabled

## Using SSL from WebSphere Application Server

To establish SSL connections from WebSphere Application Server, the applications do not need to be changed to get SSL enabled. All you need to do is prepare the keystore using the **keytool** command and set the related properties using the custom properties settings of DataSource.



Figure 4-35 show an example of settings for the WebSphere Application Server DataSource:

- ▶ `sslConnection`: true
- ▶ `sslTrustStoreLocation`: Specify the keystore location. In the example, “/home/db2inst1/keystore” is full path to the keystore named “keystore”.
- ▶ `sslTrustStorePassword`: Specify the keystore password. In the example, “passwd123” was a given for our settings.

<input type="checkbox"/>	<code>sslConnection</code>	true		false
<input type="checkbox"/>	<code>sslTrustStoreLocation</code>	/home/db2inst1/keystore		false
<input type="checkbox"/>	<code>sslTrustStorePassword</code>	*****		false

Figure 4-35 Sample datasource custom properties settings for WebSphere Application Server

### Using SSL in DB2 for z/OS DRDA Requester

Here we briefly discuss using SSL for DB2 for z/OS DRDA Requester. We will not discuss policy definitions needed for Policy Agent, but you will need add your definition for RACF and Policy Agents as discussed in “Prepare to use SSL at DB2 9 for z/OS server” on page 152.

In addition to defining AT-TLS policy, you must update the CDB table SYSIBM.LOCATIONS as follows:

- ▶ Populate the SECURE column with ‘Y’
- ▶ Populate the PORT column with secure port of the DB2 for z/OS server

**Note:** For a SSL connection, the PORT column of SYSIBM.LOCATION table must be populated with the secure port of DB2 for z/OS server, but if the SECURE column is ‘Y’ and the PORT column is blank, then DB2 uses the reserved secure port number of 448 as the default.

## 4.3.4 DataPower

The Extensible Markup Language (XML) has become the pervasive mechanism of choice for representing data among today's enterprise networks. Applications in an SOA implementation use messages with self-describing XML content to exchange information and to coordinate distributed events.

Recently, the IT industry sought to increase the ability of network devices to understand and operate upon application data.

Indeed, as XML adoption within enterprises increases, the growing number of slow-performing applications demonstrates that existing overtaxed software systems cannot support next-generation applications.

Enterprises need a platform that addresses XML's performance, security, and management requirements head-on. WebSphere DataPower® Integration appliance XI50 is a complete, purpose-built hardware platform for delivering manageable, secured, and scalable SOA solutions.

The IBM WebSphere DataPower SOA appliance portfolio includes the following elements:

- ▶ IBM WebSphere DataPower XML Accelerator XA35
- ▶ IBM WebSphere DataPower XML Security Gateway XS40
- ▶ IBM WebSphere DataPower Integration appliance XI50



For detailed explanations and scenarios, refer to *DB2 9 for z/OS: Deploying SOA Solutions*, SG24-7663.

## 4.4 Addressing dynamic SQL security concerns

Many DB2 for z/OS sites have traditionally restricted the amount of dynamic SQL that can be used, for reasons of performance and security. However, an inescapable aspect of the growing tide of application servers is that they use dynamic SQL extensively. Consequently, it is getting more difficult to keep dynamic SQL out of DB2 for z/OS.

*Squeezing the Most out of Dynamic SQL with DB2 for z/OS and OS/390*, SG24-6418, shows many techniques to manage dynamic SQL with DB2 for z/OS. Refer to that book for guidance on the wider issues of using dynamic SQL with DB2 for z/OS. It includes a section on the security implications of dynamic SQL.

Besides considerations on performance, discussed in 5.6, “Developing static applications using pureQuery” on page 211, the main security concerns with dynamic SQL are summarized here:

- ▶ It usually requires granting direct table access to primary authorization IDs or secondary authorization IDs. This exposes the risk that users do have the authorization to manipulate the data directly. By contrast, with the static SQL security model, users are granted execute authority on specific packages, which means that they can only manipulate the data using the applications that they are supposed to be using.
- ▶ The administrative burden of granting access to data objects can be much higher than granting execute authority on packages.

The static SQL security model offers major advantages to security and administration effort. What is really needed are techniques to apply the static SQL security model to dynamic SQL applications. The techniques in this section do precisely that.

**Note:** Keep in mind that network trusted context and roles also provide further levels of security.

In this section we explore the following techniques to resolve dynamic SQL security issues:

- ▶ Using DYNAMICRULES(BIND) to avoid granting table privileges
- ▶ Using stored procedures for static SQL security benefits
- ▶ Static SQL options of JDBC to realize static SQL security benefits
- ▶ Static execution of dynamic SQL to benefit from static SQL security

### 4.4.1 Using DYNAMICRULES(BIND) to avoid granting table privileges

DYNAMICRULES(BIND) provides a technique that allows dynamic SQL to be executed dynamically, but to be authorized based on the static SQL security model.

#### What it does

The DYNAMICRULES(BIND) bind option for dynamic SQL packages forces the authorization checks to be performed against the package owner, rather than the user who is executing the dynamic SQL.

## Benefits

The benefit is that users can execute dynamic SQL applications without needing table access authorities.

## How it works

All SQL, including dynamic SQL, is executed through a package in DB2. For example, if you look into the SQL contents of the package for the DB2 Command Line Processor, you can see a number of generic “PREPARE”, “DECLARE”, “OPEN”, “FETCH”, and “CLOSE” statements.

**Tip:** You can examine the SQL contents of DB2 Connect bind files using the bind file descriptor utility, as follows:

```
db2bfd -s db2clpcs.bnd
```

The DYNAMICRULES(BIND) option is an instruction to DB2 to perform authorization checking against the owner of the package, rather than the authorization ID that executes the SQL statement. It works as follows:

- ▶ The dynamic SQL package PACKX is bound using owner AUTHID1 with option DYNAMICRULES(BIND).
- ▶ The user ENDUSER1 is granted EXECUTE authority on package PACKX. GRANT EXECUTE ON PACKAGE PACKX TO ENDUSER1
- ▶ AUTHID1 has SELECT authority against TABLEX.
- ▶ At runtime, the application prepares and executes a dynamic SELECT statement against TABLEX.
  - DB2 checks that ENDUSER1 has execute privilege on the package.
  - DB2 checks that AUTHID1 currently has SELECT authority for TABLEX.

## Implementation considerations

DYNAMICRULES(BIND) is a valuable technique when the dynamic SQL application does not provide any free-form SQL processing facilities (such as SPUFI on z/OS or DB2 CLP on DB2 Connect). If the application contains a free-form SQL processor, then anybody who has execute authorization on the package, also inherits all the table authorization privileges held by the package owner.

DYNAMICRULES(BIND) can be used with dynamic SQL applications that provide free-form SQL processing facilities if you ensure that the authorization limits of the package owner are acceptable from a security viewpoint. For example, if the package owner only has select privilege on a defined set of tables, it can be acceptable to use DYNAMICRULES(BIND) and accept the possibility that users might compose their own dynamic SQL queries against that set of tables.

Multiple versions of a dynamic SQL package can be created in different collections. This would allow multiple groups of users to execute the same dynamic SQL program through different instances of the same package. For example, users from group A could be directed to the package in collection A by setting the DB2CLI.INI variable CURRENTPACKAGESET to A. By combining this technique with the DYNAMICRULES(BIND) BIND option, you can have each group running the same dynamic SQL program, with the table privileges held by the package owner of their particular instance of the package.

**Note:** This technique has an exposure if the user has the ability to edit the DB2CLI.INI file on her own workstation. There is no authorization required in DB2 for z/OS to issue SET CURRENTPACKAGESET. Hence, a user could choose to modify the DB2CLI.INI to another collection, and take advantage of the table privileges held by any other authorization ID that has bound a version of the same package.

DYNAMICRULES(BIND) does not work well for ODBC and JDBC applications, because the packages used for binding ODBC and JDBC interfaces are generic and can be used by any application using these APIs. The result would be that any user with EXECUTE privilege on the ODBC and JDBC packages would have all the privileges of the package owner.

In short, it is probably best to stick with DYNAMICRULES(RUN) for the ODBC and JDBC packages, and grant table authority to secondary authorization IDs for those users who need ODBC and JDBC access to tables.

#### 4.4.2 Using stored procedures for static SQL security benefits

Stored procedures can be called by dynamic SQL programs, and can execute their work using static SQL within the procedures, and so derive the security strengths of the static SQL model.

Stored procedures are also a good performance option for DRDA applications because they can eliminate multiple network messages by executing all the SQL within the database server environment.

Stored procedures with static SQL are also good for security reasons, since you can simply grant execute privilege on the procedure, rather than access privileges on the tables that are accessed in the procedures. An ODBC or JDBC application can issue a dynamic CALL statement and invoke a static stored procedure to run under the authority of the package owner for that stored procedure.

**Note:** DB2 for LUW and DB2 Connect have implemented the CALL statement as a fully compiled statement. It can be dynamically prepared in CLI, ODBC, JDBC, SQLJ, and embedded SQL.

Stored procedures with dynamic SQL are also good for security reasons, but they require a bit more effort to plan the security configuration.

- ▶ If you bind the package for the stored procedure with DYNAMICRULES(BIND) then the dynamic SQL in the stored procedure will also be authorized against the package owner for the dynamic SQL program.
- ▶ There are five other possible values for the DYNAMICRULES parameter (RUN, DEFINEBIND, DEFINERUN, INVOKEBIND, and INVOKERUN). Each of these other values will result in the authorization for dynamic SQL in a stored procedure being checked against an authorization ID other than the package owner. *Squeezing the Most out of Dynamic SQL with DB2 for z/OS and OS/390*, SG24-6418, contains a table that clarifies which authorization ID is used for stored procedures with dynamic SQL. Chapter 7 provides a detailed examination of the meaning of DYNAMICRULES bind parameter values. Refer to this publication for in-depth guidance on the range of possibilities that exist.

### 4.4.3 Static SQL options of JDBC to realize static SQL security benefits

The pureQuery technology and SQLJ is a database interface for Java that derives the security strengths of the static SQL model.

The primary database API for the Java application server world is JDBC. JDBC is a form of dynamic SQL, and suffers from the same security concerns as other dynamic SQL vehicles.

pureQuery or SQLJ is closely related to JDBC, and enables embedding SQL in Java methods. It offers performance and security benefits for Java applications by virtue of the fact that the DB2 implementation of static SQL.

DB2 developer domain offers many fine articles covering the benefits of pureQuery and SQLJ.

There are several articles on pureQuery but “Understanding pureQuery, Part 1: pureQuery: The IBM paradigm for writing Java database applications,” by Azadeh Ahadian gives quick introduction to the technology. It can be found at the following Web page:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0708ahadian/>

The article “Considering SQLJ for Your DB2 V8 Java Applications” by Connie Tsui does an excellent job of explaining the strengths of SQLJ in the areas of performance, security, and development simplicity. It can be found at the following Web page:

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0302tsui/0302tsui.html>

### 4.4.4 Static execution of dynamic SQL to benefit from static SQL security

We examine two situations:

- ▶ “Using pureQuery technology to benefit from static SQL security” on page 176
- ▶ “ODBC/CLI static SQL profiling to benefit from static SQL security” on page 178

We discuss a comparison at “Usage considerations” on page 181.

#### Using pureQuery technology to benefit from static SQL security

As part of IBM Data Studio pureQuery technology, you have the capability to run JDBC or .NET-based dynamic SQL application as a static application. As already addressed, you will get a benefit in security, because data access is granted to packages and not DB2 objects. Depending on your installation, you will get other benefits by making application static execution, but they are not be addressed here.

**Note:** .NET static execution capability was added as a part of IBM Data Studio pureQuery Runtime V2.1.

We briefly explain how to run SQL statements that are in a JDBC application or .NET application statically. The steps to execute dynamic SQL application using pureQuery are illustrated in Figure 4-36 on page 177. Similar steps apply to .NET applications.

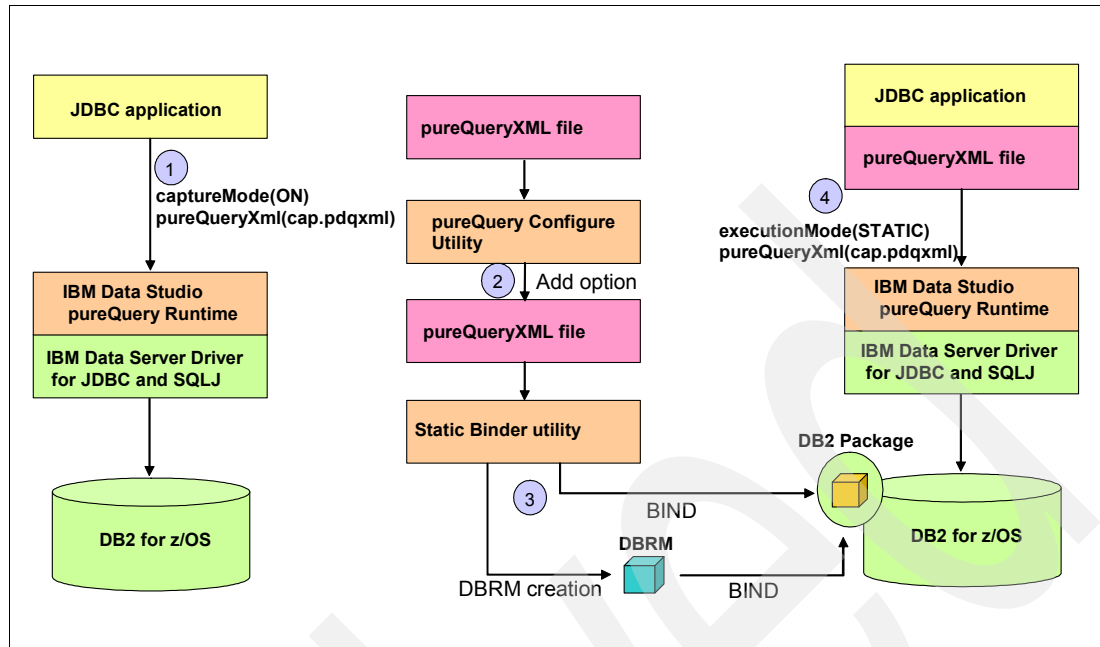


Figure 4-36 Overview of preparation steps to execute JDBC application in static mode

The basic steps to run statically are as follows:

1. Capture the SQL statements that you want to run statically.

For Java applications, set `pdqProperties` with `captureMode(ON)` and `pureQueryXml(capture.pdqxml)` to start capturing by executing application.

For .NET applications, you can pass option through connection string, such as `captureMode=ON`, `collection=COL1`, `rootPackageName=APPL1`, and `pureQueryXML=capture.pdqxml`.

2. Specify options for configuring the DB2 packages that you will create in the next step from the captured SQL statements.

For Java application, you will need to add options to the pureQuery XML file as follows:

```
java com.ibm.pdq.tools.Configure -pureQueryXml capture.pdqxml -rootPkgName
APPL1 -collection COL1
```

For .NET application, the above options are already specified so you do not need to go through this step.

3. Create and bind the DB2 packages that contain the SQL statements.

For Java application, use the static Binder utility to bind captured SQL into packages.

```
java com.ibm.pdq.tools.StaticBinder -url
jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A -username PA0L0R7 -password newpswd
-bindOptions "QUALIFIER PA0L0R7" -pureQueryXml C:\TEMP\capture.pdqxml
```

For .NET application, the bind command is:

```
db2cap bind C:\TEMP\capture.pdqxml -d DB9A -u PA0L0R7 -p newpswd
```

4. Run the JDBC application in static mode, so that the captured SQL statements run statically.

For java application, set `pdqProperties` with `executionMode(STATIC)` and `pureQueryXml(capture.pdqxml)` to execute your application in static mode.

For .NET application, set your option through connection string, using `executionMode=STATIC` and `pureQueryXML=cpature.pdqxml`.

**Note:** You can also use IBM Data Studio Developer to run your application statically. Above java application step are explained using stand-alone application from command line. For WebSphere Application Server application using JPA, static generator tool `wsdb2gen.bat` is provided to do similar steps.

### ODBC/CLI static SQL profiling to benefit from static SQL security

Static SQL profiling is a technique that allows dynamic SQL to be captured from CLI/ODBC applications executed dynamically, and bound into static SQL packages. At runtime, the dynamic SQL statements are substituted at the DB2 client with calls to a static SQL package, and so derive the security strengths of the static SQL model.

Static SQL profiling is subject to the following limitations and dependencies:

- ▶ Static SQL profiling works with any application that uses the DB2 CLI. Hence, CLI and ODBC applications are eligible.
- ▶ Static SQL profiling depends on the SQL statement text being an identical match. It does not parse the SQL statements at runtime to match them to bound SQL statements.
- ▶ Static SQL profiling does support variable predicates by using parameter markers in the SQL statements.
- ▶ If an SQL statement is not matched, it continues to execute dynamically.

To use static SQL profiling, you follow similar steps to `pureQuery`, after the program development has been completed and all the dynamic SQL statements are “fixed.”

1. Run the application and capture the dynamic SQL statements.
2. Bind the captured dynamic SQL statements to DB2.
3. Enable SQL statement matching mode.

#### **Step 1: Run the application and capture the dynamic SQL statements**

You must edit the `db2cli.ini` settings to enable `STATICSQL` capture mode. You can do this using the DB2 configuration GUI, or manually edit the `db2cli.ini` file. Example 4-33 shows a sample `db2cli.ini` file that contains the four required parameter settings for capture mode.

*Example 4-33 The `db2cli.ini` configuration for static SQL profiling capture mode*

```
C:\>more "C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY1\db2cli.ini"
```

```
[DB9A]
STATICMODE=CAPTURE
STATICPACKAGE=PA0LOR7.CAPTURE
STATICCAPFILE=C:\Temp\residency\CAPTURE1.SQL
STATICLOGFILE=C:\Temp\residency\CAPTURE1.LOG
autocommit=0
DBALIAS=DB9A
```

Pay close attention to the value of `STATICPACKAGE`.

When you run the dynamic SQL program, the SQL statements will be captured and written to file `C:\Temp\residency\CAPTURE1.SQL`. An audit trail of actions is written to `STATICLOGFILE=C:\Temp\residency\CAPTURE1.LOG`.

We tested the feature with db2cli.exe program, which executes SQL as script. It contains one SQL statement, as shown in Example 4-34.

**Tip:** The db2cli.exe is shipped with DB2 Connect or data server driver copy. For example, in DB2 Connect PE/EE for Windows, you can find db2cli.exe as:

\Program Files\IBM\SQLLIB\samples\cli\db2cli.exe

*Example 4-34 Sample CLI script used to test static profiling*

---

```
opt calldiag on
opt echo on
sqlallocenv 1
SQLAllocConnect 1 1
SQLConnect 1 DB9A -3 PAOLOR7 -3 NEWPSWD -3
SQLAllocStmt 1
SQLExecDirect 1 "SELECT COUNT(*) FROM SYSIBM.SYSTABLES" -3
fetchall 1
SQLDisconnect 1
SQLTransact 1 1 SQL_ROLLBACK
SQLDisconnect 1
SQLFreeHandle SQL_HANDLE_DBC 1
SQLFreeHandle SQL_HANDLE_ENV 1
```

---

After the program has run, it produces an SQL capture file STATICCAPFILE=C:\Temp\residency\CAPTURE1.SQL, as shown in Example 4-35.

*Example 4-35 Static profiling capture file*

---

; Captured on 2009-04-27 12.12.10.

```
[COMMON]
CREATOR=
CLIVERSION=09.02.0000
CONTOKENUR=
CONTOKENCS=
CONTOKENRS=
CONTOKENRR=
CONTOKENNC=
```

```
[BINDOPTIONS]
COLLECTION=PAOLOR7
PACKAGE=CAPTURE
DEGREE=
FUNCPATH=
GENERIC=
OWNER=PAOLOR7
QUALIFIER=PAOLOR7
QUERYOPT=
TEXT=
```

```
[STATEMENT1]
SECTNO=
ISOLATION=CS
STMTTEXT=SELECT COUNT(*) FROM SYSIBM.SYSTABLES FOR FETCH ONLY
STMTTYPE=SELECT_CURSOR_WITHHOLD
```

```
CURSOR=SQLCURCAPCSO
OUTVAR1=INTEGER,,,,TRUE,,SQL_NAMED
```

---

You may want to edit the captured SQL file to change the OWNER field under the [BINDOPTIONS] section of the captured SQL file. The captured value will be the primary authorization ID that was used to run the program when the tracing was performed. The owner should be changed to the authorization ID that will be used for authorization checking at runtime.

You may also want to pay close attention to the QUALIFIER value. If your application uses unqualified SQL, then this will be used as the qualifier by the package at bind time.

### **Step 2: Bind the captured dynamic SQL statements to DB2**

To bind the captured SQL, simply run the db2cap utility provided with the DB2 client. The syntax of the command follows:

```
db2cap [-h | -?] bind capture-file -d db-alias [-u userid [-p password]]
```

The command used in this test follows:

```
db2cap bind C:\Temp\residency\CAPTURE1.SQL -d DB9A -u paolor7 -p *****
```

It produces a package in DB2 for z/OS named CAPTURE1, in collection PAOLOR7 (as specified in the db2cli.ini parameter STATICPACKAGE=PAOLOR7.CAPTURE1).

The db2cap utility actually updates the consistency tokens in the SQL capture file to ensure a matching consistency token with DB2 for z/OS.

After binding the package you need to perform three more steps:

1. Grant execute on <collid.package> to <end\_user\_authid or group\_authid>.
2. Copy the captured SQL file to all client workstations where matching is required.
3. Edit the db2cli.ini file on those workstations to enable matching (as shown in step 3).

### **Step 3: Enable SQL statement matching mode**

To enable matching mode, the db2cli.ini file just needs to be updated to specify a STATICMODE of MATCH, as shown in Example 4-36.

*Example 4-36 db2cli.ini configuration for static SQL profiling match mode*

---

```
STATICMODE=MATCH
STATICPACKAGE=PAOLOR7.CAPTURE
STATICCAPFILE=C:\Temp\residency\CAPTURE1.SQL
TATICLOGFILE=C:\Temp\residency\CAPTURE1.LOG
autocommit=0
DBALIAS=DB9A
```

---

If the static SQL matching is successful, it is be visible through tracing, and is reported to the match mode log file. The capture log file was active during the test for both capture and match phases in this test, and is shown in Example 4-37 on page 180.

*Example 4-37 Capture - match log file*

---

```
Capture mode started. DSN=DB9A, COLLECTION=PAOLOR7, PACKAGE=CAPTURE, authorization
ID=PAOLOR7, CURRENTPACKAGESET=NULLID , CURRENTSQLID=PAOLOR7, capture
file=C:\Temp\residency\CAPTURE1.SQL.
Number of statements captured is 1.
```



CLI0008I Capture mode terminated.

**Match mode started.** DSN=DB9A, COLLECTION=PAOLOR7, PACKAGE=CAPTURE, authorization ID=PAOLOR7, QUALIFIER=PAOLOR7, OWNER=PAOLOR7.

**Number of successful distinct statement matches: 1.**

Match mode successfully completed.

---

## Usage considerations

The value of the static execution can range from being “magic” to being a “waste of effort”, depending on the ability of the application to take advantage of it.

If you are using pureQuery or static SQL profiling as a security technique, then you will want to ensure that the end users do not have privilege to access to the underlying tables. In circumstance, if SQL statements are not matched, the user receives a security SQLCODE when dynamic execution was attempted. Therefore, it is important to trace all possible SQL paths in the application.

If you are using those techniques for a performance technique, you may also want to grant access to the tables to the end users, so that statements still complete successfully, even if they were not matched, and have to be executed dynamically.

JDBC application scenarios that meet the following criteria are good candidates:

- ▶ Theoretically, all JDBC applications are eligible. pureQuery 1.2 has the following prerequisites:
  - JRE™ 1.5 or higher.
  - IBM Data Server Driver for JDBC and SQLJ, release 3.52 or higher
- ▶ Be sure application management changes will take effect
  - Security changes
  - Access path management. You will have to take consideration how many SQL statements you capture per file, which will likely become one package.
  - You should be aware of all the SQL you are going to capture. Currently you will not be able to see if all the SQL in application logic are captured or not, since technology captures SQL executed.

ODBC/CLI application scenarios that meet the following criteria are good candidates:

- ▶ Dynamic SQL with parameter markers executed through the ODBC/CLI driver.
- ▶ A relatively small number of discrete SQL statements to be matched at runtime.
- ▶ High volume application (for CPU benefits).
- ▶ Established mechanism in place to easily distribute the capture the SQL file and the db2cli.ini file to all clients (or use the thin DB2 client so that only one copy of the capture file and db2cli.ini is needed).

Archived



## Part 3

# Distributed applications

This part contains the following chapters:

- ▶ Chapter 5, “Application programming” on page 185
- ▶ Chapter 6, “Data sharing” on page 233

Archived

# Application programming

When dealing with distributed applications on various platforms, the bind options, configuration parameters, and datasource properties configured on the requester can affect your application's performance on the DB2 for z/OS server.

In this chapter we focus on remote application best practices from various requesters connecting to a DB2 for z/OS server.

This chapter contains the following sections:

- ▶ "Accessing data on a DB2 for z/OS server from a DB2 for z/OS requester" on page 186
- ▶ "Migrating from DB2 private protocol to DRDA" on page 194
- ▶ "Program preparation steps when using non-DB2 for z/OS Requesters" on page 200
- ▶ "Using the non-Java-based IBM Data Server Drivers" on page 203
- ▶ "Using the IBM Data Server Driver for JDBC and SQLJ" on page 207
- ▶ "Developing static applications using pureQuery" on page 211
- ▶ "Remote application development" on page 216
- ▶ "XA transactions" on page 225
- ▶ "Remote application recommendations" on page 229

## 5.1 Accessing data on a DB2 for z/OS server from a DB2 for z/OS requester

In this section we discuss how to connect to a DB2 for z/OS server from a DB2 for z/OS requester and recommended bind options for remote access. It is assumed that you have already set up the Communications Database (CDB) on the DB2 for z/OS requester as explained in 3.2.2, “Configuring the Communications Database” on page 86.

### 5.1.1 System-directed versus application-directed access

There are two main ways to code for distributed data from your application:

- ▶ Using explicit SQL CONNECT statements (application-directed remote access)

The SQL CONNECT statement requires that the application be aware of where the data resides, because it first has to connect to that remote database before issuing the SQL statement.

- ▶ Using three-part names/implicit DRDA (system-directed remote access)

Implicit DRDA or three-part names, you can use an alias to hide the actual location of the data from the application, making the location of the data transparent to the application.

Before we go into the details of how and when to use SQL CONNECT and three-part names statements, it is important to have a basic understanding of how connections to remote databases work from an application point of view, especially when more than one connection is involved.

#### Connection management during distributed data access

Coding for distributed data requires some information about the connecting states of your application.

Figure 5-1 on page 187 shows the details about the application and SQL connection states.

An application process can be in a connected or an unconnected state, and have zero or more SQL connections. An single SQL connection can be in one of the following four states:

- ▶ Current and held
- ▶ Current and release pending
- ▶ Dormant and held
- ▶ Dormant and release pending

An application process is initially in the *connected* state and its SQL connection is *current and held*. Issuing a successful CONNECT to another location (or implicitly through a three-part name), or a SET CONNECTION statement, makes that current SQL connection *dormant and held*. Issuing a RELEASE puts the connection in the *release pending* state. Changing a connection's status from *held* to *release pending* does not have any effect on its status as *held* or *dormant*.

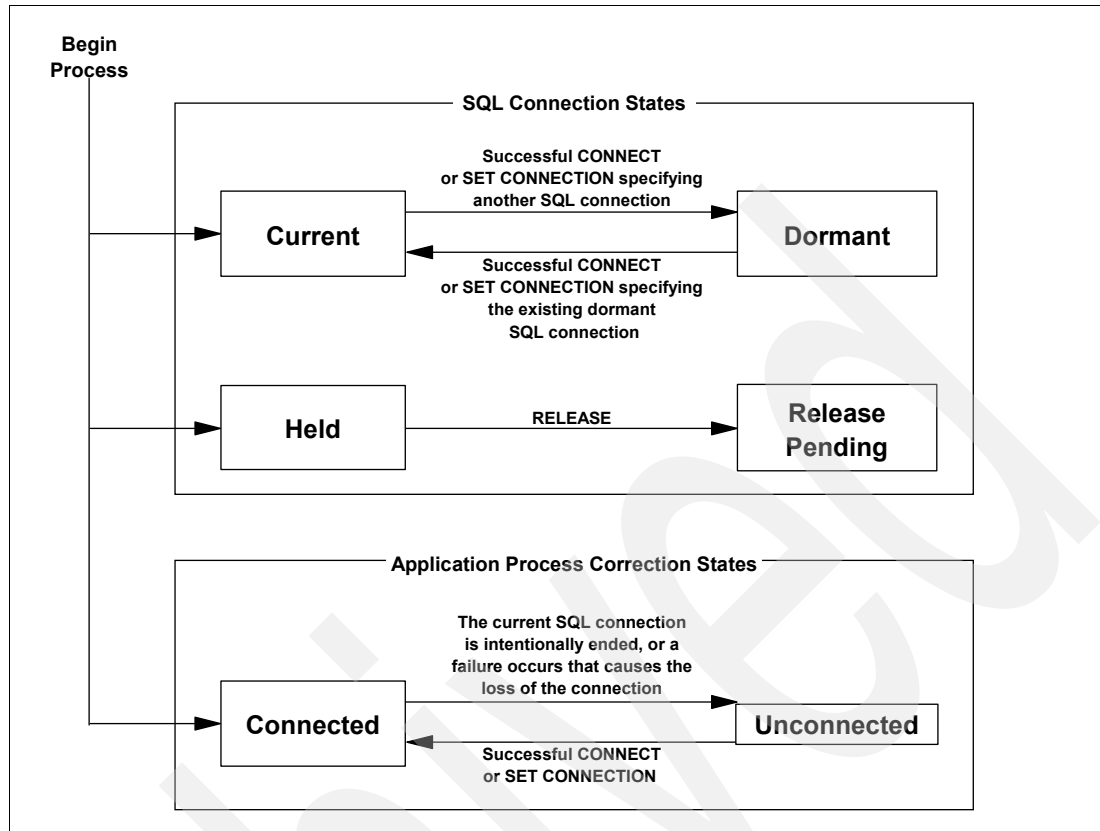


Figure 5-1 Connection management

### Three-part names

The easiest way to understand three-part table names is by looking at the syntax. We give an example in Example 5-1.

Example 5-1 Three-part table names

---

```
SELECT * FROM DB9A.SYSIBM.SYSTABLES;
```

---

The first qualifier of the three-part name is the *location name* of the server where the DML is going to be executed. The location name must be defined in the CDB. The second part is the *table/view qualifier*, and the last part is the table or view *name*. Qualifier and object name are the same as you normally use for local access.

For convenience, you can also define aliases instead of using full three-part table names. An example is shown in Example 5-2.

Example 5-2 Aliases for three-part table names

---

```
CREATE ALIAS DB9ATABLES FOR DB9A.SYSIBM.SYSTABLES;
SELECT * FROM DB9ATABLES;
```

---

Using aliases provides some form of *location transparency* because it allows you to move the data to another DBMS location without having to change your applications. When you move the data to a different DBMS in another location, you only have to drop and recreate the aliases pointing them to the new location. When you use explicit CONNECT statements, you have to change the location name in the application in all the CONNECT TO statements.

When you use three-part names, connections are implicitly established and released as you execute your SQL statements referencing them. The default protocol used to establish a connection when three-part names are used is DRDA. Starting with DB2 9 for z/OS, if you choose to override it with the DBPROTOCOL PRIVATE bind option, you receive a warning message on BIND.

When an application uses three-part names/aliases for remote objects and DRDA access, the application program must be bound at each location that is specified in the three-part names. Also, you need to define the alias at the remote site as well as at the local site for all remote objects that are accessed through aliases. For example, if your current SQL ID or authorization ID is ADMF001 and you issue the CREATE ALIAS statement in Example 5-3, it does not resolve correctly.

*Example 5-3 Incorrect alias definition*

---

```
CREATE ALIAS MYALIAS1 FOR DB9A.AUTHID2.EMPTABLE
```

---

ADMF001.MYALIAS1 is not found at DB9A because the alias refers to AUTHID2.EMPTABLE. You need to create the alias shown in Example 5-4 at the remote location DB9A.

*Example 5-4 Correct remote alias definition*

---

```
CREATE ALIAS ADMF001.MYALIAS1 FOR AUTHID2.EMPTABLE
```

---

## Explicit CONNECT statements

Another way to access distributed data is using explicit SQL CONNECT statements. In contrast with three-part names, the CONNECT statement is only supported in the DRDA protocol. If you used explicit CONNECT statements in your application program but bound your package with DBPROTOCOL(PRIVATE), DRDA is used to access the remote server.

Example 5-5 gives a basic example of using an explicit CONNECT statement.

*Example 5-5 Explicit CONNECT statement*

---

```
CONNECT TO DB9A;  
SELECT * FROM SYSIBM.SYSTABLES;
```

---

When you execute the CONNECT statement, the application connects to a (remote) server. You can hard code the location name or use a host variable. The CONNECT changes the CURRENT SERVER special register to reflect the location of the new server. After the CONNECT statement was successfully executed, any subsequent statement runs against the new location. You can return to the local DB2 subsystem by issuing the CONNECT RESET statement. It is not possible to perform a join of tables residing in two different DB2 subsystems (distributed request) unless you use the InfoSphere Classic Federation Server for z/OS.

DB2 for z/OS also supports the syntax:

```
EXEC SQL CONNECT TO :LOC USER :AUTHID USING :PASSWORD;
```

When you issue explicit CONNECT statements to remote locations, you can release these connections explicitly by issuing SQL RELEASE statements. This way, you have more flexibility over the duration of your connections. When you execute a RELEASE statement in your program, the statement does not immediately release the connection. The connection is labeled as *release-pending*, and is released at the next commit point.



The explicit RELEASE statement follows:

```
RELEASE DB9A;
```

An application can actually use two CONNECT types: Type(1) and Type(2). The two types of CONNECT statements have the same syntax, but different semantics. At precompile time, the CONNECT(1/2) option determines the type of connection that will be used for the application.

► CONNECT Type(1)

CONNECT(1) is basically a remote unit of work support. When you connect to a second system using a CONNECT(1) connection, the new CONNECT ends any existing connections of the application process, and closes any open cursors.

► CONNECT Type(2)

CONNECT(2) connections are basically to support distributed units of work. Type(2) connections do not end any existing connections or close any cursors. Connecting to another system does not do any implicit COMMIT. You must issue your COMMIT statements explicitly according to your application flow.

For detailed information about CONNECT Type(1), Type(2), and RELEASE statements, refer to *DB2 Version 9.1 for z/OS SQL Reference*, SC18-9854.

Table 5-1 summarizes the differences between the CONNECT Type(1) and CONNECT Type(2) statements.

Table 5-1 Type 1 and Type 2 CONNECT statements

Type1	Type 2
CONNECT statements can be executed only when the application process is in the connectable state. Only one CONNECT statement can be executed within the same unit of work	More than one CONNECT can be executed within a UOW. There are no rules about the connectable state.
When a CONNECT statement fails because the application is not in the connectable state, the connection state is unchanged. If the SQL connection fails for any other reason, the application process is placed in an unconnected state.	If a CONNECT statement fails, the current SQL connection is unchanged, and any subsequent SQL statements are executed at the current server, unless a failure prevents the execution of other statements by that server.
CONNECT ends any existing connections of the application process, and it closes any open cursors. No implicit commit issued.	No cursors closed, no connections ended.
A CONNECT to the current server is treated like any CONNECT(1) statement.	If the SQLRULES(STD) bind option is in effect, a CONNECT to an existing SQL connection of the application process is an error. Thus, a CONNECT to the current server is an error. For example, an error occurs if the first CONNECT is a CONNECT TO x where x is the local DB2. If the SQLRULES(DB2) bind option is in effect, a CONNECT to an existing SQL connection is not an error. Thus, if x is an existing SQL connection of the application process, CONNECT TO x makes x its current connection. If x is already the current connection, CONNECT TO x has no effect on the state of any connections.

**Note:** Programs containing CONNECT statements that are precompiled with different CONNECT precompiler options cannot execute as part of the same application process. An error occurs when an attempt is made to execute the invalid CONNECT statement.

## 5.1.2 Program preparation when DB2 for z/OS is the AR

This section covers program preparation steps including precompiler and bind options when both the requester and server are DB2 for z/OS subsystems.

Refer to *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9842 for details on program preparation and *DB2 Version 9.1 for z/OS Command Reference*, SC18-9844 for details on BIND options.

### Precompiler options

#### ► CONNECT

If you are accessing more than one server in a unit of work (DUW), you must use the CONNECT(2) precompiler option. Using CONNECT(1) explicitly only allows your programs to connect to one server at a time. The default and recommended option is CONNECT(2).

#### ► SQL

If you are accessing a non-DB2 for z/OS server, use the SQL(ALL) option. This option allows you to use any statement that is in the DRDA standard. If you are only accessing DB2 for z/OS, you can use SQL(DB2). In that case, the precompiler only accepts statements that are used by DB2 for z/OS.

### Binding your applications

When you want to access remote objects using DRDA, you must use packages at the remote site.

Example 5-6 illustrates different ways to bind, copy and deploy your remote package.

In our environment, DB9A is the local site and DB9C is the remote site.

#### *Example 5-6 Binding packages at a remote site*

---

```
-- Bind package at the remote site after shipping the DBRM
-- This is identical to a local bind
BIND PACKAGE(TESTCOL) MEMBER(TES2DCON)

-- Remote package bind running on your local system, binding DBRM TES2DCON
-- in local DBRM-lib on remote location DB9C into collection TESTCOL
BIND PACKAGE(DB9C.TESTCOL) MEMBER(TES2DCON)

-- Remote package bind running on your local system, copying package TES2DCON
-- in collection TESTCOLL on local DB2, and binding on remote location DB9C into
-- collection TESTCOL
BIND PACKAGE(DB9C.TESTCOL) COPY(TESTCOLL.TES2DCON) COPYVER(V1)

--Remote package bind running on your local system deploying 2 different versions
--of native SQL procedure package SP2DCON to remote server DB9C.

BIND PACKAGE(DB9C.TESTSPCOL) DEPLOY(PAOLOR3.SP2DCON) -
```

```
COPYVER(V1_return_4_rows) ACTION(ADD)

BIND PACKAGE(DB9C.TESTSPCOL) DEPLOY(PA0L03.SP2DCON) -
COPYVER(V2_return_3_rows) ACTION(ADD)
```

---

In DB2 9 for z/OS, BIND DEPLOY can be used to deploy a native SQL procedure to another DB2 for z/OS server. When using the DEPLOY option, only the collection ID, QUALIFIER, ACTION, COPYVER and OWNER options are allowed. This allows a functionally-equivalent procedure to be copied to another development environment in a local or remote DB2 subsystem, but does not allow other changes. Note that the access path is not copied when you use the COPY or DEPLOY options and the package will be re-optimized.

For more details on how to deploy a native SQL procedure to other DB2 for z/OS servers, refer to *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9841.

For details on package management, refer to *DB2 9 for z/OS: Packages Revisited*, SG24-7688.

Example 5-7 illustrates three different ways to bind your plan. Using wildcards gives you the flexibility to add new packages without having to rebind the plan.

---

*Example 5-7 Binding packages into the plan*

---

```
-- BIND PLAN using specific list of local and remote packages
-- Only package PKNAME in collection COLLNAME on the local system
-- and LOCNAME remote system can be executed
BIND PLAN PKLIST (COLLNAME.PKNAME,LOCNAME.COLLNAME.PKNAME)

-- BIND PLAN allowing all packages in a collection to be executed
-- All packages in collection COLLNAME on the local stem
-- and LOCNAME remote system can be executed
BIND PLAN PKLIST (COLLNAME.*,LOCNAME.COLLNAME.*)

-- BIND PLAN allowing all packages in a collection to be executed at all locations
-- All packages in collection COLLNAME can be executed on local and all remote systems
BIND PLAN PKLIST (*.COLLNAME.*)
```

---

## **BIND PACKAGE options**

Let us now have a brief look at the bind options that are specific to distributed database access.

The options are:

► **SQLERROR**

Use SQLERROR(CONTINUE) so that when there are statements that are not valid at the current server, the package is still created. Also, when binding the same package at multiple servers some references to objects may not be valid on all of the servers, because the objects do not exist there. Using this option solves this problem.

► **CURRENTDATA**

Use CURRENTDATA(NO) (default value in DB2 9 for z/OS) to force block fetch for ambiguous cursors whenever possible.

► ISOLATION LEVEL

This can have an impact on block fetch depending on the CURRENT DATA value. Isolation Level CS (Cursor Stability) is the normal and default setting - avoid using RR to allow the server to use block fetch. Refer to the table in the *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

► KEEP DYNAMIC

KEEP DYNAMIC(YES) is used to cache the SQL statements information inside the DBM1 address space, at the thread level. This option should not be confused with the dynamic statement cache, or global cache feature in DB2. When dynamic statement cache (global cache) and KEEP DYNAMIC(YES) is turned on, then DB2 keeps the prepared statements and the statement string. When only KEEP DYNAMIC(YES) is used, and the global cache is not active, then only the statement string is kept inside the DBM1 address space. Note that the use of KEEP DYNAMIC(YES) may prevent a distributed connection from being inactivated.

► SQLRULES>

Use SQLRULES(DB2) for more flexibility when binding in coding your applications, particularly for LOB data, and to improve performance.

► DEFER>

Use DEFER(PREPARE) to improve performance for dynamic SQL so that the PREPARE and EXECUTE are chained and sent together to the server. Note that this means that the application receives a SQLCODE 0 on PREPARE, even though the statement may contain invalid text or objects, and is not validated until the EXECUTE or DESCRIBE statement is executed by the application. This means that the application program has to handle SQL codes that are normally returned at PREPARE time, but can now occur at EXECUTE/DESCRIBE/OPEN time.

► RELEASE>

RELEASE(COMMIT) is the recommended option. Even if packages were bound with RELEASE(DEALLOCATE) at a DB2 for z/OS server on behalf of a DRDA client connection, DB2 for z/OS would treat them as though they were bound with RELEASE(COMMIT) during execution time.

The following options are specific to BIND PLAN and distributed access.

► DISCONNECT

The default and most flexible option is EXPLICIT. This releases all connections in release pending state at commit time. This means that you must use RELEASE statements in your applications to end the connections. Other values are AUTOMATIC and CONDITIONAL. AUTOMATIC releases all connections at commit, even those that were not released by the application. CONDITIONAL ends remote connections at commit, except when open, with-hold cursors are in use by the connection.

► CURRENTSERVER

Determines the location to connect to before running the plan. The column CURRENTSERVER in SYSIBM.SYSPPLAN records this value and the special register. CURRENT SERVER receives this value when the plan is allocated. Use this value when you want an application to use the data at another DB2 server without changing the application. Avoid using this keyword when there are explicit CONNECT statements in your application. An implicit type 1 connection is used with this option and causes any explicit CONNECT statements in the application to be treated as type 1, even if the application was precompiled with option type 2.

Figure 5-2 on page 193 illustrates the use of packages and their execution within a local and remote DB2 subsystem when accessing distributed data.

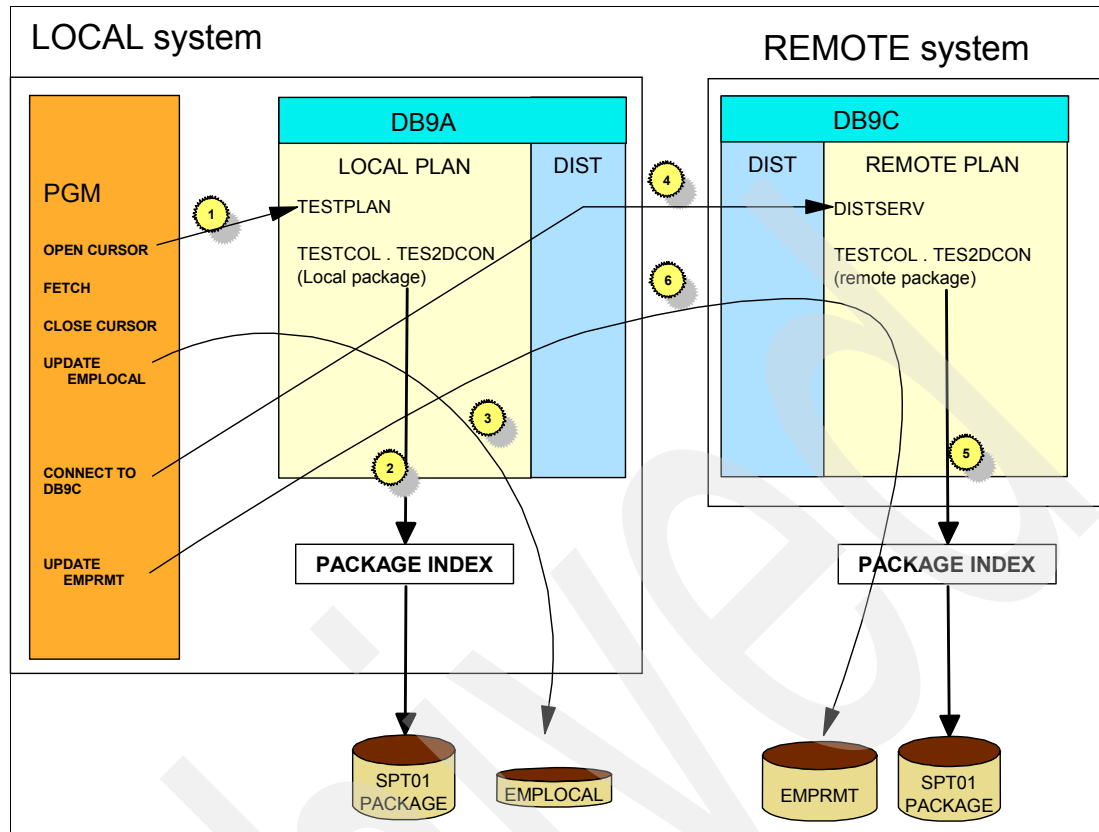


Figure 5-2 Execution of remote packages

The flow in Figure 5-2 is as follows:

1. When the first SQL statement is executed, the local plan (TESTPLAN) is used.
2. Package (TESTCOL.TES2DCON) is allocated on the local system (DB9A).
3. The local update to the EMPLOCAL table is executed.
4. Connect to the remote DB2 system DB9C
5. When executing at a remote system using DRDA, all packages run under the same plan called DISTSERV. The remote package TESTCOL.TES2DCON is looked up and loaded from the directory of the remote system (DB9C).
6. The update on the remote table (EMPRMT) is executed as a true static SQL statement on DB9C.

### 5.1.3 Using DB2 for z/OS as a requester going outbound to a non-DB2 for z/OS server

When using DB2 for z/OS requester and going outbound to a non-DB2 for z/OS server such as DB2/LUW or DB2 for i, you need to set up your CDB tables on the DB2 for z/OS server to such that you can access the remote server through TCP/IP as documented in Chapter 3, "Installation and configuration" on page 69.

This configuration is described in 2.2, "DB2 for LUW and DB2 for z/OS" on page 37.

**Restriction:** It is not possible to use SNA to connect to a non-DB2 for z/OS server, you must use TCP/IP.

In addition to binding the packages against the local DB2 for z/OS, you need to bind against the remote server. In this case SAMPLE is the LUW database to which we are connecting. In your application program, you can either use explicit CONNECT to SAMPLE or a three-part name statement that makes a connection through implicit DRDA. Remember that when CDB outbound translation is in effect, you need to ensure that translated ID corresponding to the the BIND OWNER (not QUALIFIER) has sufficient authority to bind and execute packages on the remote server.

Example 5-8 shows a sample program that can be used to remote bind SPUFI packages against the DB2 LUW server SAMPLE. For details, refer to the following Web page:

[http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.inst/db2z\\_runspufiatremotesite.htm](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.inst/db2z_runspufiatremotesite.htm)

*Example 5-8 Binding remote SPUFI packages*

---

```
//PAOLO44B JOB (999,POK),REGION=5M,MSGCLASS=X,CLASS=A,
//          MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*JOBPARM S=SC63
/*-----
/* DRDA REDBOOK --> BIND REMOTE SPUFI PACKAGES IN AIX BOX
/*-----
//JOB LIB DD DISP=SHR,DSN=DB9C9.SDSNEXIT
//          DD DISP=SHR,DSN=DB9C9.SDSNLOAD
//DSNTIRU EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
        DSN SYSTEM(DB9A)
        BIND PACKAGE(SAMPLE.REMAIXCS) MEMBER(DSNESM68) -
          ACTION(ADD) ISO(CS) CURRENTDATA(YES) -
          LIBRARY('DB9A9.SDSNDBRM')
        BIND PACKAGE(SAMPLE.REMAIXRR) MEMBER(DSNESM68) -
          ACTION(ADD) ISOLATION(RR) -
          LIBRARY('DB9A9.SDSNDBRM')
        BIND PACKAGE(SAMPLE.REMAIXPUR) MEMBER(DSNESM68) -
          ACTION(ADD) ISOLATION(UR) -
          LIBRARY('DB9A9.SDSNDBRM')
END
```

---

## 5.2 Migrating from DB2 private protocol to DRDA

Private protocol has been deprecated in DB2 9 for z/OS and will be removed in a future release of DB2 due to the following reasons:

- ▶ Private protocol is only used by DB2 for z/OS.
- ▶ Networks are rarely homogeneous.
- ▶ Private protocol has not been functionally enhanced since DB2 Version 5.
- ▶ DRDA's support for data blocking and its improved performance make it the preferred vehicle for remote data access.

DB2 V6 enhanced the DRDA protocol with the support of three-part names and aliases. This function allowed to set the default of the DBPROTOCOL BIND option to PRIVATE protocol when binding packages or plans of applications utilizing three-part name references. Starting with DB2 9, it is no longer possible to change the default to private protocol. However, it is still possible to specify DBPROTOCOL(PRIVATE) when performing a BIND, and the request completes with a warning message DSNT226I indicating that this option is no longer recommended.

Because you should be prepared for the eventual removal of private protocol support, DB2 for z/OS provides a catalog analysis tool (DSNTP2DP) to be used on DB2 9 catalogs. This tool is provided with the DB2 installation JCL in the form of a REXX program. This new program searches the system's DB2 catalog tables to determine all private protocol dependencies known to DB2 in existing bound applications. From this search a series of bind jobs that specify DBPROTOCOL(DRDA) are automatically created.

### 5.2.1 DB2 performance trace to show private protocol use

By running a specific performance trace (IFCID 157) during execution windows of opportunity, you get a record of the packages and DBRMs that are executing private protocol statements. The trace records produced would have information such as plan name, package name, DBRM name, section number, remote location name, statement type, and SQL statement before and after alias resolution. With this information, you can create lists of packages that should be bound in preparation for DB2's elimination of Private Protocol. This is not a new trace, but rather is a performance trace of specific IFCIDs.

Because DRDA requires packages at remote sites, all programs currently using private protocol must have packages bound to the remote sites referenced in those applications. One way to determine the remote sites that are referenced in currently running programs is to run the performance trace for IFCIDs 157 to obtain this information. A START TRACE command to start these traces is shown here:

```
START TRACE(PERFM) CLASS(30) IFCID(157) DEST(GTF)
```

You can use the IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS (OMEGAMON PE) is enhanced to dump the trace records for DB2 Version 8 as well as for DB2 9.

For DB2 V8, you can download from the DB2 for z/OS Exchange section of the IBM developerWorks® Web site an unsupported version of DSNTP2DP that can be used against DB2 V8 catalogs.

### 5.2.2 The PRIVATE to DRDA REXX migration tool: DSNTP2DP

To help you convert your plans and packages from using private protocol to DRDA protocol, DB2 provides the private to DRDA protocol REXX tool, DSNTP2DP, which scans your catalog and generates the necessary commands to convert all objects that have a remote location, private protocol dependency to DRDA. We recommend that you run the DBRM conversion before running the private protocol tool, to lessen the chance of overlaying existing packages or deleting DBRMs. Refer to *DB2 9 for z/OS: Packages Revisited*, SG24-7688, for details on DBRM conversion.

You can tailor the generated output from the tool and run it at your discretion. Use job DSNTIJPDP, which is customized during migration, to invoke DSNTP2DP.

A package or plan has a remote location private protocol dependency only when the tool can extract from the catalog remote location dependency information that is related to a plan or package. Just having the DRPROTOCOL column of the catalog tables that manage plans and packages set to a value of 'P' (Private) does not mean that the plan or package has a remote location private protocol dependency. However, packages and plans that were bound with the DBPROTOCOL PRIVATE option will not be allowed in a future release of DB2.

**Important:** Use the option ACTION(REPLACE) RETAIN® for binding plans if using DB2-based security. This option preserves EXECUTE privileges when you replace the plan. If ownership of the plan changes, the new owner grants the privileges BIND and EXECUTE to the previous owner. The RETAIN option is not the default. If you do not specify RETAIN, everyone but the plan owner loses the EXECUTE privilege (but not the BIND privilege). If plan ownership changes, the new owner grants the BIND privilege to the previous owner.

The syntax and options for the tool DSNTP2DP are shown in Figure 5-3.

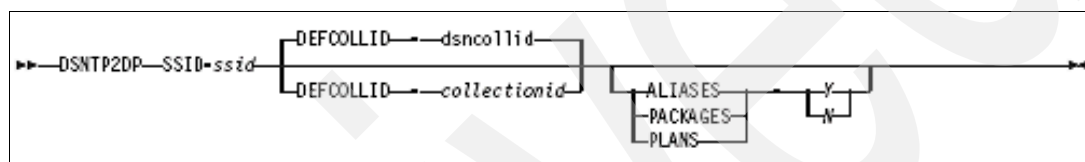


Figure 5-3 Options for tool DSNTP2DP

The tool has the following three parameters:

- ▶ Subsystem ID of DB2 to be examined (mandatory parameter) specified as SSID=ssid  
This first parameter is mandatory so that DSNTP2DP can connect to the DB2 subsystem.
- ▶ Default collection name (optional parameter) specified as DEFCOLLID=collectionid  
The second parameter is the default collection name to be used in the generated BIND commands where a collection name cannot be determined. If this parameter is not specified, then the tool assumes a default collection name of DSNCCOLLID.
- ▶ Run options (optional parameters) ALIASES=Y/N, PACKAGES=Y/N, PLANS=Y/N  
The run parameters trigger which processing the tool performs. If none of the run option parameters are specified, then all catalog objects are examined by the tool (that is, a value of Y is assumed for processing the plans, packages, and aliases in the catalog). If any of the run option parameters are specified and the value assigned to a particular run option is an N, then the processing for those catalog objects controlled by that run option is not performed.

There are two catalog tables that have a database protocol (DBPROTOCOL) flag, SYSPLAN and SYSPACKAGE. The catalog analysis tool uses this flag to extract the potential packages, DBRMs, or plans to be converted to DRDA from private protocol. Commands are only generated for those packages, DBRMs, or plans, that have a remote location dependency.

The output of the DSNTP2DP REXX exec with ALIASES=Y contains SQL statements to create aliases for the remote locations. You can execute these statements using DSNTPE2 or DSNTPE4, or any application that can process SQL statements, including CREATE ALIAS, CONNECT TO, RELEASE, and COMMIT.



## Determine whether there are existing packages that are bound with DBPROTOCOL(PRIVATE)

Bind those packages with DBPROTOCOL(DRDA) at the correct locations. Running the DSNTP2DP exec with PACKAGES=Y, which is run as part of job DSNTIJP, provides output that assists with this task.

### Aliases

If the ALIASES run option is Y, then all aliases are examined. For each alias that references a remote object and the fully qualified name of the remote object (for example, AUTHID.OBJECT) does not match the fully qualified name of the alias (for example, CREATOR.NAME) a command is generated to connect to the remote system and create the necessary two-part name alias at the remote location.

### Packages

If the PACKAGES run option is Y, then all the packages in the catalog are examined next. For each local package that has a DBPROTOCOL set to P and a dependency on a remote location, a command will be generated to REBIND the local PACKAGE with the DBPROTOCOL(DRDA) option. This PACKAGE will be used as the source for the next generated BIND PACKAGE COPY commands against each server location where the PACKAGE has a remote dependency. When binding any of the packages to remote location servers, SQLERROR CONTINUE will also be specified to ensure that the package is bound to the remote locations, because not all of the statements within the package may actually reference objects in the remote servers.

Figure 5-4 shows the commands generated to create the local package and to copy the local package to the remote systems.

```
DSN SYSTEM(DB9A)
* This file can be used as input to TSO batch.
* Note: for each target location referenced in remote
* bind requests, a different userid (other than the one
* running the TSO batch job) may be used to access the
* target location depending on the configuration within
* this subsystem's CDB. That userid must either have
* SYSADM authority on the target syssubsystem or it must
* have suitable privileges and authorities to bind the
* package into the collections of the target location.
REBIND PACKAGE(LI682COLLID.LI682C) -
  DBPROTOCOL(DRDA)
BIND PACKAGE(DB9A.LI682COLLID) COPY(LI682COLLID.LI682C) -
  OPTIONS(COMPOSITE) -
  OWNER(PAOLOR3) QUALIFIER(PAOLOR3) -
  DBPROTOCOL(DRDA) SQLERROR(CONTINUE)
REBIND PACKAGE(LI682COLLID.LI682D.(V1R1M0)) -
  DBPROTOCOL(DRDA)
BIND PACKAGE(DB9C.LI682COLLID) COPY(LI682COLLID.LI682D) -
  COPYVER(V1R1M0) OPTIONS(COMPOSITE) -
  OWNER(PAOLOR3) QUALIFIER(PAOLOR3) -
  DBPROTOCOL(DRDA) SQLERROR(CONTINUE)
```

Figure 5-4 Sample output for packages

## Plans

If the PLANS run option is set to Y, then the tool examines all the plans in the catalog. Regardless of the setting of the PACKAGES run option, no further packages in the catalog are examined as part of this phase of the tool's processing. Thus, this phase of the tool only generates actions to convert all the DBRMs that are bound directly into plans that have a remote location dependency.

For each DBRM that is bound directly into a plan that has a remote dependency, a BIND PACKAGE command will be generated that will bind the DBRM as a PACKAGE locally within a specified collection (or default collection DSNCOLLID) using BIND PACKAGE parameters that can be extrapolated from the current PLAN parameters and any parameter values from the corresponding SYSDBRM row but ensure that DBPROTOCOL(DRDA) is specified. The source for the DBRM will be obtained from the PDSNAME column of the corresponding SYSDBRM row. The next set of generated commands will be BIND PACKAGE COPYs with the specified collection (or default collection DSNCOLLID) specified against each server location to be accessed by the DBRM package, and the source will be the package just created. In addition, the binding of these packages to remote location servers will also have the SQLERROR CONTINUE option specified.

As a final step of this phase, a BIND PLAN is generated to replace the existing PLAN with a new PKLIST if none was previously present, or an updated PKLIST if one was previously present and DBPROTOCOL(DRDA) is specified. The BIND PLAN command's PKLIST parameter must now include the local and remote collections.

**Important:** If you run this tool against your DBRM-based plans using private protocol, you create statements that will create packages, both locally and remotely. When the BIND PLAN statement is generated, it will not have any DBRMs bound directly into the plan, and all access will be through the packages listed in the PKLIST.

The tool generates the commands that should be performed to make a plan or package use DRDA protocols when accessing remote locations. These commands with the appropriate JCL are stored in a file that can then be tailored for the environment.

Figure 5-5 on page 199 shows the commands generated to create the local packages from DBRM-based plans and to BIND the local plans with the remote packages.

```

DSN SYSTEM(DB9A)
* This file can be used as input to TSO batch.
* Note: for each target location referenced in remote
* bind requests, a different userid (other than the one
* running the TSO batch job) may be used to access the
* target location depending on the configuration within
* this subsystem's CDB. That userid must either have
* SYSADM authority on the target sysssubsystem or it must
* have suitable privileges and authorities to bind the
* package into the collections of the target location.
BIND PACKAGE(DSN COLLID) MEMBER(PGM04) -
  LIBRARY('PAOLOR3.TEMP.DBRM1') -
  OWNER(PAOLOR3) QUALIFIER(PAOLOR3) -
  ...
  DBPROTOCOL(DRDA) SQLERROR(CONTINUE)
BIND PACKAGE(DB9C.DSN COLLID) COPY(DSN COLLID.PGM04) -
  OPTIONS(COMPOSITE) -
  OWNER(PAOLOR3) QUALIFIER(PAOLOR3) -
  DBPROTOCOL(DRDA) SQLERROR(CONTINUE)
BIND PLAN(FINPRIV1) ACTION(REPLACE) RETAIN -
  ACQUIRE(USE) CACHESIZE(256) DISCONNECT(EXPLICIT) -
  OWNER(PAOLOR3) QUALIFIER(PAOLOR3) -
  VALIDATE(RUN) -
  ISOLATION(CS) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  NODEFER(PREPARE) -
  DEGREE(1) -
  DYNAMICRULES(RUN) -
  REOPT(NONE) -
  KEEP DYNAMIC(NO) -
  ENCODING(37) -
  IMMEDIATE WRITE(NO) -
  SQLRULES(DB2) -
  PKLIST(DSN COLLID.*,DB9C.DSN COLLID.*) -
  DBPROTOCOL(DRDA)

```

Figure 5-5 Sample output from PLANS data set

## Complete the conversion

After you have run the DSNTDP2DP REXX exec and have the SQL created for the aliases, and the bind commands necessary for both packages and plans, execute these commands. You can use DSNTDP2, DSNTDP4 or your favorite SQL processor to create the aliases. Run the BIND commands for the packages and plans using JCL you would use to run any other bind statements.

You might also want to replicate plans and packages that need to be converted into a development environment, run the tool there, exercise the new binds, test the code and then deliver tested scripts to production.

**Attention:** For DB2 9 for z/OS, obtain the PTF for APAR PK78553 which provides a more complete version of DSNTDP2DP. For DB2 for z/OS V8, a version of DSNTDP2DP is available through developerWorks:

[http://www.ibm.com/developerworks/exchange/dw\\_entryView.jspa?externalID=213&categoryID=32](http://www.ibm.com/developerworks/exchange/dw_entryView.jspa?externalID=213&categoryID=32)

## 5.3 Program preparation steps when using non-DB2 for z/OS Requesters

In this section, we discuss the program preparation steps when using DB2 Connect or any of the IBM Data Server Drivers/Clients as Application Requesters. Remember that Command Line Processor (CLP) is bundled with DB2 Connect and the IBM Data Server Clients/Runtime clients but not available when using the thin Data Server Drivers. When CLP is not available you can use the DB2Binder utility (described in 5.3.2, “Using the DB2Binder utility to bind packages used by the Data Server Drivers” on page 203) to bind packages required by the data server drivers on the DB2 for z/OS Server.

### 5.3.1 Connecting and binding packages from DB2 CLP

Provided that you have already cataloged your database either using CCA (Client Configuration Assistant) or from the command line, you can connect to a DB2 for z/OS server from DB2 CLP using the CONNECT...USER...USING syntax, or as shown in Example 5-9 where you are prompted for your password.

*Example 5-9 Connecting from DB2 CLP*

```
C:\Program Files\IBM\SQLLIB\BIN>db2 connect to DB9A user paolor3
Enter current password for paolor3:
```

#### Database Connection Information

Database server	= DB2 z/OS 9.1.5
SQL authorization ID	= PAOLOR3
Local database alias	= DB9A

**Tip:** We recommend that you explicitly bind the packages specified in DDSCMversusLST with BLOCKING ALL and SQLERROR CONTINUE with a user ID that has BINDADD authority in the NULLID collection. If you do not bind explicitly, another user connected to the target DB2 for z/OS subsystem with the right authority running a query will cause an implicit rebind. The DDSCMversusLST packages will be implicitly bound without BLOCKING.. All subsequent users of the gateway will experience poor query performance because rows will be returned on a one row per block basis.

You can specify any generic bind option following the generic keyword. You need to be in the BND directory to perform the BIND. You need to switch back to the BIN directory to execute remote SQL.

For details, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.ci.doc/doc/t0006343.html>

Example 5-10 shows how to bind packages used by DB2 Connect at a remote DB2 for z/OS server.

*Example 5-10 Binding DB2 Connect packages on a remote DB2 for z/OS*

```
C:\Program Files\IBM\SQLLIB\bnd>db2 bind @ddcmvs.lst blocking all sqlerror continue
```

```
LINE      MESSAGES FOR ddcsmvs.lst
```

```
-----
SQL0061W  The binder is in progress.
```

```

LINE   MESSAGES FOR db2clist.bnd
-----
SQL0038W  The bind option SQLERROR CONTINUE has been
          activated since it is required when binding this DB2-supplied
          list file to DB2/MVS, SQL/DS, or OS/400.
SQL0038W  The bind option SQLERROR CONTINUE has been
          activated since it is required when binding this DB2-supplied
          list file to DB2/MVS, SQL/DS, or OS/400.

LINE   MESSAGES FOR db2clpcs.bnd
-----
2958    SQL0408N  A value is not compatible with the data type of its
          assignment target. Target name is "XMLVAL".  SQLSTATE=42821

LINE   MESSAGES FOR db2clprp.bnd
-----
2958    SQL0408N  A value is not compatible with the data type of its
          assignment target. Target name is "XMLVAL".  SQLSTATE=42821

LINE   MESSAGES FOR db2clpur.bnd
-----
2958    SQL0408N  A value is not compatible with the data type of its
          assignment target. Target name is "XMLVAL".  SQLSTATE=42821

LINE   MESSAGES FOR db2clprs.bnd
-----
2958    SQL0408N  A value is not compatible with the data type of its
          assignment target. Target name is "XMLVAL".  SQLSTATE=42821

LINE   MESSAGES FOR ddcsmvslst
-----
SQL0091N  Binding was ended with "0" errors and "6" warnings.

```

C:\Program Files\IBM\SQLLIB\bnd>

---

Example 5-11 shows the use of command db2bfd (bind file display) that you can use to confirm what defaults were used when binding a particular package.

The options for db2bfd are as follows:

- ▶ -b = display bind parameters
- ▶ -s = display statements in the .bnd file
- ▶ -v = display information about the host variables

*Example 5-11 Using command db2bfd*

---

C:\Program Files\IBM\SQLLIB\bnd>db2bfd -b -s -v db2clist.bnd

db2clist.bnd: Header Contents

Header Fields:

Field	Value
releaseNum	0x800
Endian	0x4c

```

numHvars      25
maxSect       37
numStmt       39
optInternalCnt 4
optCount      10

```

Name	Value
Isolation Level	Uncommitted Read
Creator	"NULLID "
App Name	"SYSSTAT "
Timestamp	"SYSLVL01:User defined timestamp"
Cntrlreqd	Yes
Sql Error	Continue
Block	Block All
Date	ISO
Time	ISO
Validate	Bind

\*\*\* All other options are using default settings as specified by the server \*\*\*

```

db2clist.bnd: SQL Statements = 39
<<SQL statements and host variables not shown>>

```

---

Once you have connected and bound the packages, you can issue remote SQL interactively from DB2 CLP as shown in Example 5-12.

*Example 5-12 Executing remote SQL interactively from DB2 CLP*

```

C:\Program Files\IBM\SQLLIB\BIN>db2 select count(*) from sysibm.systable
-----
5904

1 record(s) selected.

```

---

You can also add a set of SQL statements to a file and use the db2 -f <inputfilename> facility to execute them from CLP. You can create a native SQL procedure and call the procedure using literals from DB2 CLP as shown in Example 5-13. If you need to use host variables in your CALL, then you have to use CLI.

*Example 5-13 Creating and calling a native SQL procedure from DB2 CLP*

```

C:\Program Files\IBM\SQLLIB\BIN>db2 call nishu.write_to_log22(1,'parm2','parm3',
'parm4')
RT: IT WAS OK

"WRITE_TO_LOG22" RETURN_STATUS: 0

```

---

### 5.3.2 Using the DB2Binder utility to bind packages used by the Data Server Drivers

The DB2Binder utility is shipped with the IBM Data Server Driver for JDBC and SQLJ and is used to bind the packages used at the DB2 for z/OS by the driver. It also grants EXECUTE authority on the packages to PUBLIC. You can also use this utility to rebind DB2 packages that are not part of the IBM Data Server Driver for JDBC and SQLJ. See Example 5-14.

---

#### *Example 5-14 Using the DB2Binder utility*

---

```
C:\DDF\test\javatests>java com.ibm.db2.jcc.DB2Binder -url
jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A -user paolor3 -password yyyy
Binder performing action "add" to "jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A" under
collection "NULLID":
<Other messages from Binder not shown>
```

---

The default collection used is NULLID, but if you wanted to bind your packages with non-default bind options, you can explicitly specify the collection and bind options you want to use. You can then use the `setCurrentPackagePath()` property to specify which package you want to use at execution time. You can also specify bind options as a property for the DB2Binder class as shown in Example 5-15 where the `defaultQualifierName` is set for native SQL procedure package that will be deployed to a remote site.

---

#### *Example 5-15 Bind options as a property for the DB2Binder class*

---

```
Properties bndOpts = new Properties();

connection = (com.ibm.db2.jcc.DB2Connection) DB2simpleDatasource.getConnection ();
System.out.println ("Connection successful");
bndOpts.setProperty("defaultQualifierName", "PAOLOR3");
com.ibm.db2.jcc.DB2Binder binder = new com.ibm.db2.jcc.DB2Binder ();
binder.deployPackage ("DB9A", "PAOLOR4", "PAOLOR4", "PSM756", "V1", bndOpts,
connection);
```

---

## 5.4 Using the non-Java-based IBM Data Server Drivers

As described at 2.3, "IBM Data Server Drivers and Clients as requesters" on page 40, you can use one of several data server drivers, clients, or runtime clients to access the DB2 for z/OS server.

### 5.4.1 Using the IBM Data Server Driver for ODBC and CLI

In this section we will briefly discuss recommended configuration parameters when using IBM Data Server Driver for ODBC and CLI.

Example 5-16 shows a C program snippet that creates a connection to the DB2 for z/OS server and sets the AutoCommit Connection attributed to false.

---

#### *Example 5-16 Connecting to DB2 for z/OS through CLI*

---

```
rc=SQLAllocHandle(SQL_HANDLE_DBC, henv1, &hdbc1);
if (rc != SQL_SUCCESS) goto dberror;

// location/uid/pwd in passed in as parms
rc=SQLConnect(hdbc1, loc1, SQL_NTS, loc1uid, SQL_NTS, loc1pwd, SQL_NTS);
```

```

if (rc != SQL_SUCCESS) goto dberror;

// sample code to toggle autocommit
// autocommit=yes is cli default, so set=off for this connection.
// add explicit commit/rollback if you turn autocommit off.
rc=SQLSetConnectAttr( hdbc1,SQL_ATTR_AUTOCOMMIT,(void*) SQL_AUTOCOMMIT_OFF,
SQL_NTS);
if (rc != SQL_SUCCESS) goto dberror;

```

---

Example 5-17 shows the db2dsdriver.cfg file that can be used at the client to enable sysplex workload balancing with automatic client reroute (ACR) as well as enabling direct XA transactions<sup>1</sup> without having to go through a DB2 Connect Server.

**Tip:** When you use the data server drivers, you can add the remote server details directly to this configuration file without having to catalog your database.

*Example 5-17 The db2dsdriver.cfg file*

```

<configuration>
  <DSN_Collection>
    <dsn alias="DB9C_DIR" name="DB9C" host="wtsc63.itso.ibm.com" port="38320"/>
    <!-- Long aliases are supported -->
  </dsn>
</DSN_Collection>
<databases>
  <database name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
    <WLB>
      <parameter name="enableWLB" value="true"/>
      <parameter name="maxTransports" value="100"/>
    </WLB>
    <ACR>
      <parameter name="enableACR" value="true"/>
    </ACR>
  </database>
</databases>
  <parameters>
    <parameter name="enableDirectXA" value="true"/>
  </parameters>
</configuration>

```

---

## 5.4.2 Using the IBM Data Server Driver Package in a .NET environment

In June 2009, IBM announced the IBM Data Server Driver package which includes drivers for the .NET and OpenSource environments. In this section we will focus on the usage of the Data Server Driver Package in a .NET environment which extends DB2 data server support for the ADO.NET interface. The IBM Database Development Add-Ins enable you to develop .NET applications for IBM data servers using Microsoft Visual Studio. You can also use the Add-Ins to create database objects such as indexes and tables, and develop server-side objects, such as stored procedures and user-defined functions.

<sup>1</sup> XA transactions come from the X/Open group specification on distributed, global transactions.



.NET applications can be developed in Visual Basic or C# and to connect to the DB2 for z/OS server, you would need to provide the IP address, port, user ID, and password in a configuration file.

Figure 5-6 shows a sample .NET application developed using Microsoft Visual Studio.

```
Imports System
Imports System.Data
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Web
Imports IBM.Data.DB2
Imports System.Data.Common

Namespace BART_GETCARPOOLERS1

    <WebService(), _
        WebServiceBinding()> _
    Public Class BART_GETCARPOOLERS1

        <WebMethod()> _
        Public Overridable Function [Select](ByVal MYCITY_param As String) As DataSet
            Dim db2Connection1 As IBM.Data.DB2.DB2Connection = New
IBM.Data.DB2.DB2Connection
            Dim db2DataAdapter1 As IBM.Data.DB2.DB2DataAdapter = New
IBM.Data.DB2.DB2DataAdapter
            Dim db2SelectCommand1 As IBM.Data.DB2.DB2Command = New IBM.Data.DB2.DB2Command
            'db2Connection1.ConnectionString =
"database=DB9A;userid=rajesh;server=wtsc63.itso.ibm.com:12347;password=rathnam2"
            db2Connection1.ConnectionString =
ConfigurationManager.ConnectionStrings("TEST").ConnectionString
            db2Connection1.ClientUser = ConfigurationManager.AppSettings("TEST1")

            db2DataAdapter1.SelectCommand = db2SelectCommand1
            db2DataAdapter1.SelectCommand.CommandType = CommandType.StoredProcedure
            db2DataAdapter1.SelectCommand.CommandText = "BART.GETCARPOOLERS1"
            db2DataAdapter1.SelectCommand.Parameters.Add("MYCITY",
DB2Type.LongVarChar).Value = MYCITY_param
            db2DataAdapter1.SelectCommand.Connection = db2Connection1
            Dim ds As System.Data.DataSet = New System.Data.DataSet
            Try
                db2DataAdapter1.Fill(ds)
            Catch ex As DB2Exception
                Throw ex
            Finally
                db2Connection1.Close
            End Try
            Return ds
        End Function
    End Class
End Namespace
```

Figure 5-6 .NET application code sample

Figure 5-7 shows the associated configuration file.

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="TEST1" value="ABCD"/>
  </appSettings>
  <connectionStrings>
    <add name="TEST"
connectionString="database=DB9A;userid=rajesh;server=wtsc63.itso.ibm.com:12347;password=rathnam2"
providerName="IBM.Data.DB2"/>
  </connectionStrings>
  <system.web>
    <!-- Set compilation debug="true" to insert debugging symbols into the compiled page.
Because this affects performance, set this value to true only during development. -->
    <compilation debug="true">
      <assemblies>
        <add assembly="IBM.Data.DB2, Version=9.0.0.2, Culture=neutral,
PublicKeyToken=7C307B91AA13D208"/>
      </assemblies>
    </compilation>
    <!-- The <authentication> section enables configuration of the security authentication mode
used by ASP.NET to identify an incoming user. -->
    <authentication mode="Windows"/>
    <!-- The <customErrors> section enables configuration of what to do if/when an unhandled
error occurs during the execution of a request. Specifically, it enables developers to configure
html error pages to be displayed in place of a error stack trace.<customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm"><error statusCode="403" redirect="NoAccess.htm" /><error
statusCode="404" redirect="FileNotFound.htm" /></customErrors>-->
  </system.web>
</configuration>
```

Figure 5-7 .NET configuration file

For more information, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.ms.doc/doc/c0010960.html>

### 5.4.3 db2cli.ini and db2dsdriver.cfg

When you are using the IBM Data Server Drivers, you need to configure the db2dsdriver.cfg file. When using DB2 Connect, the Data Server Client, or Data Server Runtime Client, you still use db2cli.ini. When using CLI, you can also specify Statement or Connection attributes instead of the db2cli.ini keyword. Typically, this is the CLI keyword prefixed with SQL\_ATTR. For example, DeferredPrepare corresponds to SQL\_ATTR\_DEFERRED\_PREPARE.

If the CLI/ODBC configuration keywords set in the db2cli.ini file conflict with keywords in the SQLDriverConnect() connection string, the SQLDriverConnect() keywords take precedence.

**Tip:** Syntax errors in db2dsdriver.cfg are silently ignored. If you want to make sure your settings are getting picked up, update your database configuration using diaglevel 4 through CLP (or manually update db2cli.ini if using the thin data server drivers where CLP is not available) and check db2diag.log for error messages.

Table 5-2 lists the db2cli.ini/db2dsdriver.cfg configuration parameters that are relevant from a distributed perspective. Notice that the names of certain keywords are different between db2dsdriver.cfg and db2cli.ini and some do not have a direct equivalent (N/A).

For recommended parameters when using Workload Balancing, refer to Table 6-4 on page 254 (CLI Driver) and Table 6-5 on page 258 (DB2 Connect Server).

Table 5-2 db2cli.ini and db2dsdriver.cfg configuration parameters

db2dsdriver.cfg parameter	Equivalent CLI keyword	Description	Default/Recommended value (1 is ON, 0 is OFF)
AllowDeferredPrepare	DeferredPrepare	Combines the PREPARE and EXECUTE requests	1/1
N/A	OptimizeForNRows	Appends the Optimize for N Rows clause to each SQL	Not appended/Use when fetching large amounts of data to enable extra blocks
DisableAutoCommit	AutoCommit	Commits every statement by default	AutoCommit is 1 by default (DisableAutoCommit is 0 by default)/Turn AutoCommit OFF and use explicit COMMITs.
DisableCursorHold	CursorHold	Cursors persist past COMMIT	CursorHold is 1 by default (DisableCursorHold is 0 by default)/ When possible, set CursorHold to 0 to allow the connection to go inactive
NumRowsOnFetch	BlockForNRows	Specifies number of rows returned per block	Default is to return as many rows as can fit in a block which is also recommended.
EnableLobBlockingOnFetch	BlockLobs	Specifies if LOBs should be blocked	0/1
N/A	StreamGetData	Enables progressive streaming for LOBs (Statement or connection property only)	Default varies based on which driver or client is used/1
LOBCacheSize	LOBCacheSize	Specifies threshold up to which LOBs can be inlined	None/12K
enableDirectXA	N/A	Allows direct XA connection from the data server drivers.	0/1 when you need two-phase commit
QueryTimeoutInterval	QueryTimeoutInterval	Indicates how long the driver should wait between checks to see if the query has completed.	5 seconds/ Set to a value that is larger than the SQL_ATTR_QUERY_TIMEOUT to prevent timeouts from occurring at the specified interval.

**Important:** The precedence rule is in the following order: application, db2cli.ini, db2dsdriver.cfg.

## 5.5 Using the IBM Data Server Driver for JDBC and SQLJ

As described in Chapter 1, “Architecture of DB2 distributed systems” on page 3, The IBM Data Server Driver for JDBC and SQLJ provides Type 4 and Type 2 connectivity. We focus on the Type 4 driver in this section because it can be used to connect to a DB2 for z/OS server

through DRDA. The Type 4 driver is now delivered in the jcc3 and jcc4 streams. The jcc4 stream requires JDK™ 1.6 to be installed. You also need to customize and run the DSNITJMS job that creates stored procedures and tables required by the Type 4 driver. Refer to the following Web pages for information:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.java.doc/doc/t0024156.html>

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.java.doc/doc/c0052041.html>

To know which version of the driver you are using, use the command in Example 5-18.

*Example 5-18 Determining the driver version*

---

```
C:\DDF\test>java com.ibm.db2.jcc.DB2Jcc -version
IBM Data Server Driver for JDBC and SQLJ 4.8.23
```

---

### 5.5.1 Connecting to a DB2 for z/OS server using the Type 4 driver

You can use either the `DriverManager` or the `DataSource` interface to obtain a connection to the database server. Here is an example using the `DriverManager.getConnection()` interface where the connection properties are embedded in the application program. Example 5-19 shows the usage of the `getConnection()` interface to obtain a connection to the DB2 for z/OS server.

*Example 5-19 Using the getConnection()*

---

```
String url = "jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A";
java.util.Properties prop = new java.util.Properties();
prop.put("user", user);
prop.put("password", password);
prop.put("driverType", "4");
conn1 = DriverManager.getConnection(url,prop);
```

---

It is possible to create and use a `DataSource` object in the same application similar to the `DriverManager` interface. This method does not provide portability. The recommended way to use a `DataSource` object is for your system administrator to create and manage it separately, using WebSphere Application Server or some other tool. It is then possible for your system administrator to modify the data source attributes and you do not need to change your application program. Example 5-20 shows the use of the `DataSource` interface to obtain a connection to the DB2 for z/OS server.

*Example 5-20 Connecting to DB2 for z/OS through the DataSource interface*

---

```
DB2SimpleDataSource dataSource = new com.ibm.db2.jcc.DB2SimpleDataSource();
dataSource.setServerName (servername);
dataSource.setPortNumber (Integer.parseInt(port));
dataSource.setDatabaseName (databasename);
dataSource.setUser (user);
dataSource.setPassword (password);
dataSource.setDriverType (4);
conn1 = dataSource.getConnection();
```

---

For details on connections using the `DriverManager` or `DataSource` interfaces, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.java.doc/doc/cjvjdcon.html>

To learn more about using WebSphere to deploy DataSource objects, refer to the following Web page:

<http://www.ibm.com/software/webservers/appserv/>

In general, the default properties enabled for the Type 4 driver are designed to maximize distributed performance. If you want to change either the configuration properties that have driver-wide scope, or the Connection/Datasource properties that are application-specific, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.java.doc/doc/rjvdsprp.html>

**Tip:** Refer to Table 6-4 on page 254 for recommended settings to enable Sysplex Workload Balancing

## 5.5.2 Coding static applications using SQLJ

The main difference between using JDBC and using SQLJ to access DB2 is that JDBC always uses dynamic SQL. SQLJ can use dynamic SQL, but through the customization process can be set up to use static SQL instead. This normally means better run-time performance. The customization process (using `db2sqljcustomize`) can also be used to check the syntax of the SQL statements used in the application, which reduces the possibility of runtime errors.

**Tip:** If you want to make sure you are getting static execution for your packages, set the `db2.jcc.sqljUncustomizedWarningorException` property to 1 (warning) or 2 (exception). The default is 0, which means it will switch to dynamic SQL if there were any errors encountered during the customization process.

Refer to the following Web page for a complete list of configuration properties:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.java.doc/doc/cjvcfgpr.htm>

Figure 5-8 shows a simple application that access the DB2 for z/OS server using SQLJ. It was developed using the IBM Data Studio Developer.

```
package com.ibm.sqljdemo;

import java.sql.*;

#sql iterator EmployeeInfo(String, String, String, String, String, String, String, String, String);
#sql iterator DepartNum(String);
#sql context ctx;

public class DatabaseAccessUsingSQLJ {

    public static String listDepartmentNames(Connection con, ServletOutputStream os) {
        StringBuffer departmentListPage = new StringBuffer("ABC");

        DepartNum results;
        String department = null;
        try
        {
            ctx thisCTX = new ctx(con);
            departmentListPage.setLength(0);
            departmentListPage.append("<H3> Department </H3>");
            #sql [thisCTX] results=(SELECT distinct workdept FROM WORFDEMO.EMPLOYEE);

            while (true)
            {
                #sql {FETCH :results INTO :department};
                if (results.endFetch())
                {
                    break;
                }
                departmentListPage.append("<br>" + department);
            }

            departmentListPage.append("<br>----- End -----<br>");
            results.close();
            os.println(departmentListPage.toString());
        }
        catch (Exception e)
        {
            System.out.println("Exception:" + e.toString());
        }
        return departmentListPage.toString();
    }
}
```

Figure 5-8 SQLJ application sample

Figure 5-9 shows the db2sqljcustomize step that is necessary for static execution of SQLJ packages.

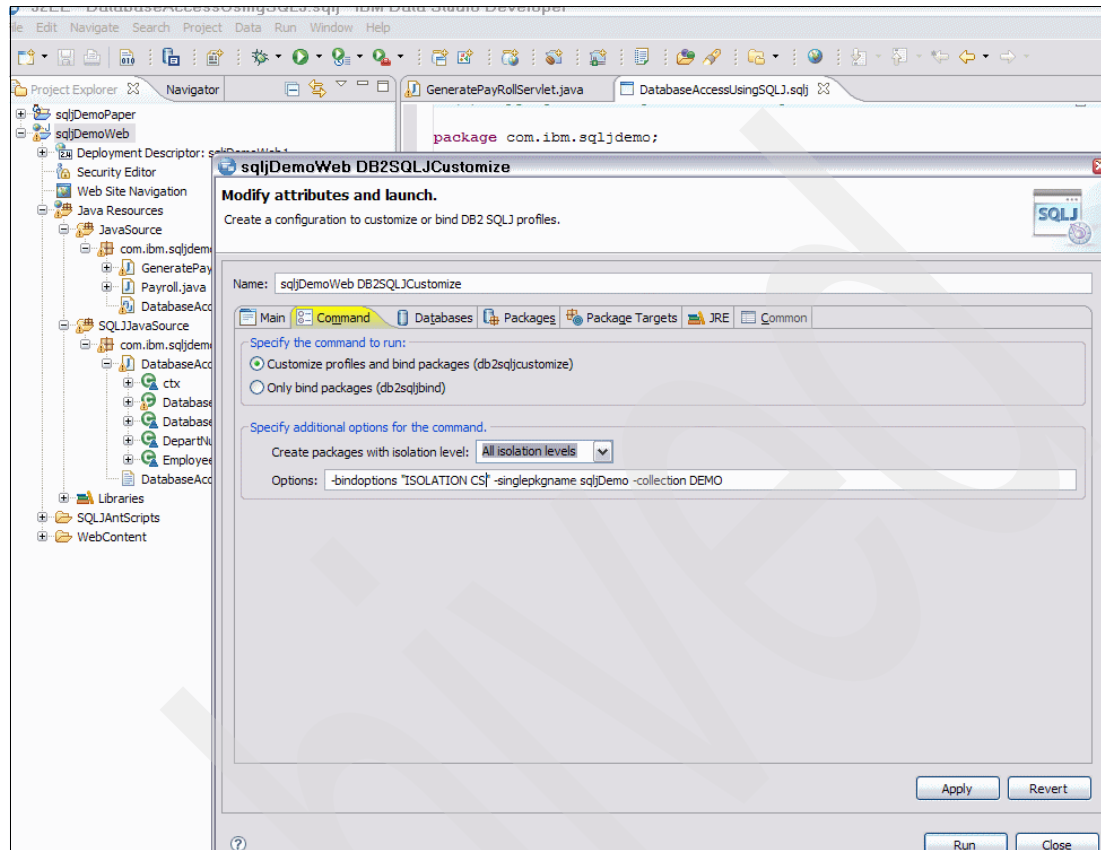


Figure 5-9 Customizing and binding an SQLJ application

## 5.6 Developing static applications using pureQuery

pureQuery is a high performance Java data access platform focused on simplifying the tasks of developing and managing applications that access data. It consists of tools, API, and a runtime engine. pureQuery allows you to capture SQL from existing JDBC (dynamic) applications and allows you to switch from dynamic to static SQL execution. This allows for improved performance, predictability, and security. It is also transparent as there is no need for source code. As shown in Figure 5-10 on page 212, when using pureQuery you will still need to use the Type 4 driver as the DRDA AR to connect to the DB2 for z/OS server.

For security considerations with pureQuery, see “Using pureQuery technology to benefit from static SQL security” on page 176.

For more information, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp>

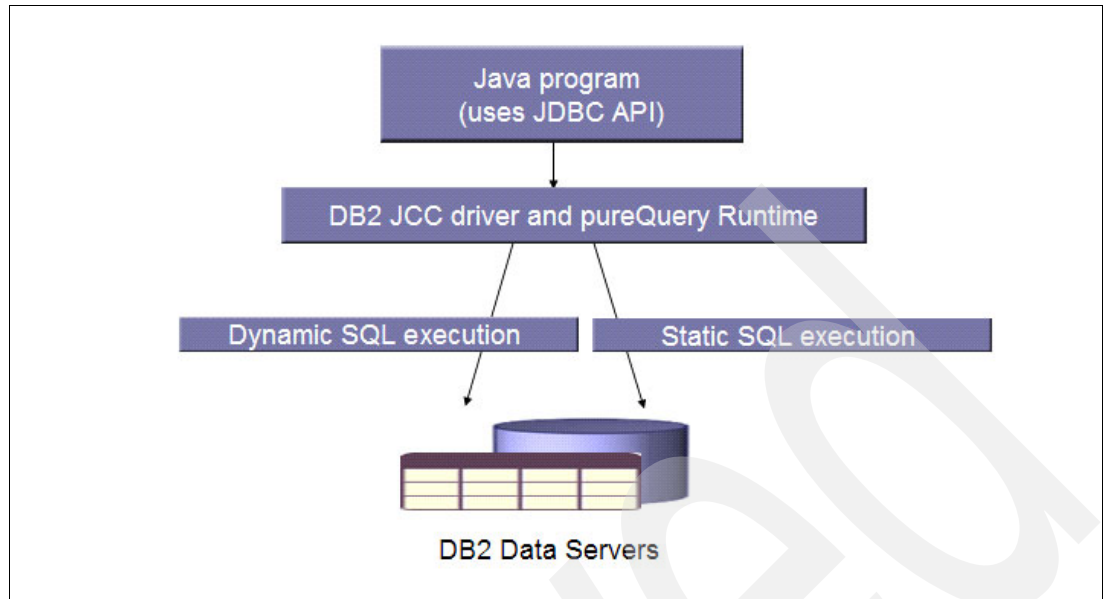


Figure 5-10 pureQuery runtime

### 5.6.1 When should you use pureQuery?

You might find the functions of pureQuery beneficial for developing or modifying applications.

#### Developing new Java applications

You will benefit from static SQL performance and security, developer tooling, and enhanced diagnostic capabilities.

#### Modifying existing JDBC applications and frameworks

If your dynamic SQL cache hit ratio is below 80%, consider using client optimization to test the advantages of static SQL.

#### Modifying existing SQLJ applications

If you are using SQLJ you are already getting the performance benefits of static SQL. But you could still benefit from new API features provided by pureQuery such as heterogeneous batch updates that allow you to combine several SQL statements into a single network call. Also it is simpler to bind and deploy with pureQuery than with SQLJ.

For more information, refer to the following Web pages:

- ▶ Data Studio Information Center (part of Integrated Data Management)  
<http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp>
- ▶ No Excuses Database Programming for Java  
[http://www.ibm.com/developerworks/data/library/dmmag/DBMag\\_2008\\_Issue2/NoExcuseSDB/index.html](http://www.ibm.com/developerworks/data/library/dmmag/DBMag_2008_Issue2/NoExcuseSDB/index.html)



## 5.6.2 pureQuery programming styles

When you use pureQuery, you can choose between the inline-method style and the annotated-method programming style. The inline style is similar to the JDBC programming style except for being simpler and faster to code. It was designed to reduce the repeated programming tasks familiar to the JDBC programmer, as well as to provide an API that tools could easily use to tie in data access development with Java development. The coding is referred to as "inline" because of the way SQL statements are defined in the application.

The annotated method coding style has the additional goal of maximizing configurability and security for the resulting pureQuery application. It was developed in response to customer demand for a named query programming interface for data access that was similar to Java Persistence API (JPA), only simpler, quicker to code, and capable of supporting static execution when required.

It is outside the scope of this book to go into the details of these two styles. Refer to the articles at the following Web pages for more information about pureQuery programming techniques:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0804lamb/>  
<http://www.ibm.com/developerworks/data/library/techarticle/dm-0804vivek/>  
<http://www.ibm.com/developerworks/data/library/techarticle/dm-0808rodrigues/>

## 5.6.3 pureQuery client optimization

Client optimization, a key enhancement in the 1.2 release, makes it possible to take advantage of static SQL for existing JDBC applications, without modifying, or even having access to the source code. This capability also works on Java applications that use a framework for persistence, such as Hibernate™. With client optimization, you execute the application while the pureQuery runtime is in capture mode, typically as part of a use case test validation that exercises various program execution paths. The DBA can then bind the captured SQL into packages at the database server. Subsequent executions of the application allow the pureQuery runtime to execute the captured SQL statically. In this way, client optimization gives applications the performance and security benefits of static SQL, without the need to modify or recompile the application.

To convert existing dynamic SQL applications to static, you need to follow this 4-step process:

1. Capture. Traps SQL statements from existing dynamic applications and records them into a capture file as illustrated by Example 5-21.

---

### *Example 5-21 Capturing dynamic SQL statements*

---

```
prop.setProperty ("pdqProperties",  
    "captureMode ON, executionMode DYNAMIC ,allowDynamicSQL true, pureQueryXml  
    abc_pdq.xml");
```

---

2. Configure. Specifies collection, package name and version about the target package and sets them to the capture file as shown in Example 5-22.

---

### *Example 5-22 Configuring target packages*

---

```
C:\DDF\test\javatests>java com.ibm.pdq.tools.Configure -pureQueryXml c:\ddf\test  
\javatests\abc_pdq.xml -rootPkgName CHECK -collection NULLID
```

---

3. Bind. StaticBinder reads the capture file configured in the previous step and binds packages to the target DB2.

Example 5-23 shows some of the enhanced error reporting features available in the StaticBinder utility in the 2.1 release such as -showDetails and -grant

---

*Example 5-23 Using the StaticBinder utility*

---

```
C:\DDF\test>java com.ibm.pdq.tools.StaticBinder -url jdbc:db2://wtsc63.itso.ibm
.com:12347/DB9A;emulateParameterMetaDataForZCalls=1; -grant "grantees(GAURAV,
MANOJ, JAIJEET)" -showdetails true -user paolor3 -password yyyy -bindOptions
"VALIDATE RUN" -pureQueryXml C:/ddf/test/abc_pdq.xml
```

The StaticBinder utility is beginning to bind the pureQueryXml file 'C:/ddf/test/abc\_pdq.xml'.

Following Exception(s)/Warning(s) were reported for package : PKGA1 at BIND STEP

```
Error Code : 204, SQLSTATE : 01532
Message : PAOLOR3.TABLE33 IS AN UNDEFINED NAME. SQLCODE=204, SQLSTATE=01532, DRI
VER=4.8.23
SQL Statement: drop table table33
SQL Locator : 1
```

The StaticBinder utility successfully bound the package 'PKGA1' for the isolation level UR.

Executed successfully : GRANT EXECUTE ON PACKAGE "C0151"."PKGA1" TO GAURAV, MANOJ, JAIJEET

<Similar messages for isolation levels CS, RR and RS not shown>  
Displaying the -showDetails results:

```
Number of packages input : '2'
Number of statements input : '5'
Number of DDL statements : '2'
Number of packages for which isBindable is false : '0'
Number of statements for which isBindable is false : '0'
Number of root packages bound : '2'
Number of statements bound: '5' (for each isolation level specified)
The number of statements in each package are as follows:
    Package : 'PKGA', statements= '2'
    Package : 'PKGB', statements= '3'
```

- 
4. Run. Run your applications statically with the capture file when execution mode is static. See Example 5-24.

---

*Example 5-24 Running the static application*

---

```
prop.setProperty ("pdqProperties", "captureMode OFF, executionMode STATIC,
pureQueryXml abc_pdq.xml");
```

---

Figure 5-11 illustrates this process.

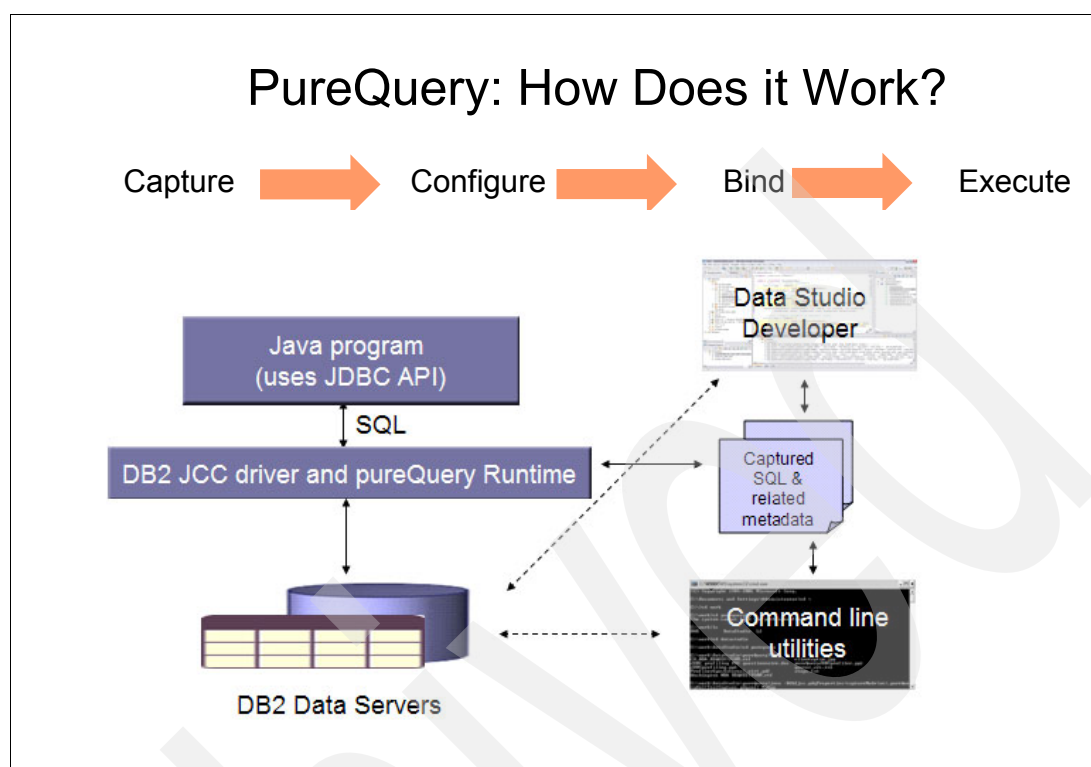


Figure 5-11 The internal workings of pureQuery

Table 5-3 highlights the differences between static and dynamic SQL applications.

Table 5-3 Comparing pureQuery dynamic and static SQL

Feature	Dynamic SQL (pureQuery, JDBC)	Static SQL (pureQuery, SQLJ)
<b>Performance</b>	Can approach static SQL performance with help from dynamic SQL cache. Cache misses are costly.	All SQL parsing, catalog access, done at BIND time. Fully optimized during execution.
<b>Access path reliability</b>	Unpredictable: Any prepare can get a new access path as statistics or host variables change	Guaranteed: locked in at BIND time. All SQL available ahead of time for analysis by EXPLAIN.
<b>Authorization</b>	Privileges handled at object level. All users or groups must have direct table privileges: Security exposure, and administrative burden.	Privileges are package based. Only administrator needs table access. Users/Groups have execute authority. Prevent non-authorized SQL execution.
<b>Monitoring, problem determination</b>	Database View is of the JDBC or CLI package: No easy distinction of where any SQL statement came from.	Package View of applications makes it simple to track back to the SQL statement location in the application.
<b>Capacity planning, forecasting</b>	Difficult to summarize performance data at program level.	Package Level Accounting gives program view of workload to aid accurate forecasting.
<b>Tracking dependent objects</b>	No record of which objects are referenced by a compiled SQL statement.	Object dependencies registered in database catalog.

## 5.7 Remote application development

In this section, we discuss some of the distributed performance topics from an application programming perspective. We have primarily used the Type 4 driver to illustrate examples but have mentioned the CLI driver as well as applications connecting from a DB2 for z/OS requester where there are significant differences.

### 5.7.1 Limited block fetch

With limited block fetch, the DB2 for z/OS server attempts to fit as many rows as possible in a query block. DB2 transmits the block of rows over the network. Data can also be pre-fetched when the cursor is opened without needing to wait for an explicit fetch request from the requester. As a result, using limited block fetch can significantly decrease the number of network transmissions.

**Attention:** DRDA supports the concept of exact blocking versus flexible blocking. All DRDA requesters at SQLAM 7 and higher can handle flexible blocks, which means a block can be expanded to include a complete row or SQL rowset (when multi-row fetch is used) and partial rows/rowsets are not returned to the requester. We limit our discussion to flexible blocking. Although the default block size is 32 K, due to this potential expansion of blocks, the DB2 9 for z/OS server may send a query block up to 10 MB in size when rowsets are involved.

Example 5-25 shows a Java application code snippet where block fetch is used to retrieve data from SYSIBM.SYSTABLES.

*Example 5-25 Block fetching in a Java program*

```
void test () throws SQLException
{
    Statement stmt1 = conn1.createStatement();
    SQLWarning warning = stmt1.getWarnings();
    if ( null != warning )
        System.out.println("Warning: "+warning.toString());
    ResultSet rs1 = stmt1.executeQuery("SELECT DBID, OBID FROM SYSIBM.SYSTABLES");
    int row = 0;
    System.out.println("Running next().....");
    while (rs1.next()) {
        row = rs1.getRow();
        rs1.getInt(1);
        rs1.getInt(2);
    }

    rs1.close();
    stmt1.close();
}
```

#### Conditions for block fetch to occur

In order for DB2 for z/OS server to use block fetch, you need to use a read-only unambiguous cursor.

**Attention:** Prior to DB2 for z/OS V8, you will see one IFCID 59 record cut for every row fetched. Due to performance enhancements in V8, you will now only see one IFCID 59 record per block fetched.

For detailed information about when DB2 for z/OS server decides to use block fetch, refer to the following Web page:

[http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.perf/db2z\\_ensureblockfetch.htm](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.perf/db2z_ensureblockfetch.htm).

### Extra blocks

DRDA supports the concept of extra blocks, where multiple query blocks can be returned to the requester in response to a single FETCH request. The number of extra blocks returned is computed as the minimum of what the requester asked for and what the DB2 for z/OS server will allow to be returned (EXTRASRV). The default value and maximum value for EXTRASRV is 100. In the case of a z/OS requester, the maximum number of extra blocks requested is controlled by EXTRAREQ. The default and maximum value for EXTRAREQ is 100. The actual number of blocks returned depends on the data and on either the OPTIMIZE FOR m ROWS (m) value or the QRYROWSET value.

### OPTIMIZE for n ROWS and FETCH FIRST n ROWS

To trigger extra blocks to be returned, you need to use either of the above clauses. Both clauses may also be used to influence access path selection.

The FETCH FIRST n ROWS clause can be used to limit the number of rows returned to the application. Once n ROWS are fetched and end of data is reached, the cursor qualifies for fast implicit close.

The OPTIMIZE for m ROWS clause is used to control both access path selection and DRDA blocking.

OPTIMIZE for 1 ROW has special meaning to the optimizer as it indicates that sort should be avoided if possible when choosing an access path. But to avoid sending too many blocks to a requester, when m is less than 4, DB2 returns 16 rows per block but uses the original m for access path selection.

The size of the row data and the size of the query block will still determine how many rows will fit in a block. The OPTIMIZE FOR m ROWS and FETCH FIRST n ROWS values can influence the maximum number of rows that the server might return for a single CNTQRY. These rows could be spread out over several query blocks. If you used a small value for m, that can potentially allow for a smaller number of rows to be fetched, hence making for a faster turnaround if the application only wants a small number of rows returned at a time. A large m value can potentially allow for extra query blocks to flow, decreasing the number of network transmissions.

**Important:** FETCH FIRST n ROWS and OPTIMIZE for m ROWS are not used for DRDA blocking when the cursor is a rowset or scrollable cursor. For scrollable cursors and rowset cursors DB2 uses the DRDA QRYROWSET parameter to determine the number of rows fetched in a block.

## 5.7.2 Multi-row FETCH

Multi-row FETCH was introduced in DB2 for z/OS V8 and the performance advantages of using multi-row FETCH over single row FETCH have been clearly established. DSNTEP4 uses multi-row FETCH by default.

Example 5-26 shows a PL/I application code snippet that retrieves a rowset of 64 rows.

*Example 5-26 Retrieving a rowset*

---

```
EXEC SQL DECLARE SPC2 INSENSITIVE SCROLL CURSOR
      WITH ROWSET POSITIONING
      WITH RETURN
      FOR
      SELECT NAME, CREATOR FROM DB9A.SYSIBM.SYSTABLES;

EXEC SQL FETCH NEXT ROWSET FROM C2 FOR 64 ROWS
      INTO :OUTVAR1, :OUTVAR2;
```

---

The Type 4 driver (version 3.57 and higher) automatically uses multi-row fetch for scrollable cursors. This means you can get the performance advantages of multi-row fetch without making any application changes. The `useRowsetCursor` property is set to true by default and the `DB2BaseDataSource.enableRowsetSupport` property if set, can override the value of the `useRowsetCursor` property.

Example 5-27 shows a Java application where multi-row fetch is used implicitly.

**Tip:** `setFetchSize()` can be used to specify the number of rows per block for scrollable rowset cursors. If a forward-only cursor is used, `setFetchSize()` is ignored by the Type 4 driver.

*Example 5-27 Implicit multi-row fetching in a Java program*

---

```
void test () throws SQLException
{
    Statement stmt1 = conn1.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                             ResultSet.CONCUR_READ_ONLY);

    stmt1.setFetchSize (64);
    ResultSet rs1 = stmt1.executeQuery("SELECT DBID, OBID FROM"
    + "SYSIBM.SYSTABLES");
    int row = 0;
    while (rs1.next()) {
        row = rs1.getRow();
        rs1.getInt(1);
        rs1.getInt(2);
    }

    rs1.close();
    stmt1.close();
}
```

---

A single FETCH statement from a rowset cursor might encounter zero, one, or more conditions. If the current cursor position is not valid for the fetch orientation, a warning occurs and the statement terminates. If a warning or non-terminating error (such as a bind out error) occurs during the fetch of a row, processing continues. In this case, a summary message is returned for the FETCH statement, and additional information about each fetched row is available with the GET DIAGNOSTICS statement. The SQLERRD3 that is part of the SQLCA only contains the number of rows returned when multi-row fetch is used.

When using the Type 4 driver, set the property shown in Example 5-28 to obtain extended diagnostic information from the DB2 for z/OS server.

*Example 5-28 Setting a property to enable extended diagnostic*

```
prop.put("extendedDiagnosticLevel","241");
```

### 5.7.3 Understanding the differences between limited block FETCH and multi-row FETCH

Table 5-4 summarizes the differences between limited block FETCH and multi-row FETCH from a DB2 9 for z/OS server perspective.

*Table 5-4 Comparing limited block FETCH and multi-row FETCH*

Limited block FETCH	Multi-row FETCH
Supported only for distributed applications through DRDA.	Supported for local as well as distributed applications through DRDA
To limit the number of rows returned to the application, use FETCH FIRST n ROWS, otherwise a block is filled to capacity or till end of result table is reached.	Number of rows fetched is limited using explicit FOR N ROWS syntax or through setFetchSize() JDBC API.
Non-atomic	May be atomic or non-atomic.
It is possible to send extra blocks when using FETCH FIRST n ROWS or OPTIMIZE for n ROWS	The QRYROWSET parameter is used for rowset cursors.
Prefetching is possible at OPEN time without waiting for an explicit FETCH request.	Prefetching is never done because the rowset size is only specified at FETCH time by the requester.

### 5.7.4 Fast implicit CLOSE and COMMIT of cursors

The DB2 for z/OS server attempts to close cursors implicitly whenever possible to avoid additional network flows that are required when the requester has to initiate a close.

DB2 uses fast implicit close when the following conditions are true:

- ▶ The query retrieves no LOBs
- ▶ The query retrieves no XML data.
- ▶ The cursor is not a scrollable cursor.
- ▶ The cursor is declared with FETCH FIRST n ROWS clause and N rows have been fetched
- ▶ One of the following conditions is true:
  - The cursor is declared WITH HOLD, and the package or plan that contains the cursor is bound with the KEEP DYNAMIC(YES) option.
  - The cursor is declared WITH HOLD and the DRDA client passes the QRYCLSIMP parameter set to SERVER MUST CLOSE, SERVER MUST CLOSE AND MAY COMMIT or SERVER DECIDES
  - The cursor is not defined WITH HOLD.

**Tip:** You will not see an IFCID 66 trace record cut when the cursor is implicitly closed by the server.

## Implicit COMMIT enhancement when using the Type 4 driver

Although we recommend that AutoCommit be turned off for distributed applications, there are certain cases where it is necessary to turn it on. The additional commit request and reply flows used to honor AutoCommit result in degraded performance as compared to when AutoCommit is not enabled. With the JCC Type 4 driver level 3.57.82/4.7.85 shipped with DB2 for LUW 9.7 and the DB2 for z/OS APAR PK68746, the server may commit after implicitly closing the cursor upon SQLSTATE 02000 (SQLCODE +100). The JCC fix is included in the following releases:

- ▶ In V9.7 (Cobra) GA
- ▶ As a special build on top of V9.1 fixpack 4 (3.6.98)
- ▶ Planned to be in the official release of V9.1 fixpack 8

The JCC property `queryCloseImplicit` can be used to enable or disable this feature.

The possible values for `queryCloseImplicit` follow:

- ▶ 0: NOT\_SET to JCC. The jcc driver will choose a default depending on the driver and server level (DB2 for z/OS requester uses 0 as the default, which means server decides)
- ▶ 1: Implicit close only
- ▶ 2: Disable implicit close
- ▶ 3: Implicit close + commit. The 9.7 jcc driver will choose this value as the default value only when targeting z/OS server version 1.10 or DB2 for z/OS server version 9 NFM with APAR PK68746 applied.

## 5.7.5 Multi-row INSERT

Similar to multi-row fetch, multi-row INSERT is supported for both local applications and distributed applications over DRDA. Example 5-29 shows a sample PL/I application that performs multi-row insert using a dynamic cursor.

*Example 5-29 Multi-row INSERT in a PL/I program*

---

```
DCL MYSTR1 CHAR(900) VARYING;
DCL MYSTR2 CHAR(100) VARYING;
DCL MYSTR3 CHAR(900) VARYING;

DCL HCHAR1(3) CHAR(3);
DCL HCHAR2(3) CHAR(1);
DCL HCHAR3(3) CHAR(10);
DCL INT1 BIN FIXED(15);

HCHAR1(1) = 'ABC';
HCHAR1(2) = 'ABC';
HCHAR1(3) = 'ABC';
HCHAR2(1) = 'A';
HCHAR2(2) = 'A';
HCHAR2(3) = 'A';
HCHAR3(1) = 'ABCDEFGH1J';
HCHAR3(2) = 'abcdefghij';
HCHAR3(3) = 'abcdefghij';
INT1 = 3;

MYSTR2 = 'FOR MULTIPLE ROWS';
MYSTR1 = 'INSERT INTO MYTABLE VALUES(?,?,?)';
```



```
EXEC SQL PREPARE STMT1 ATTRIBUTES :MYSTR2 FROM :MYSTR1;
EXEC SQL EXECUTE STMT1 USING :HCHAR1, :HCHAR2, :HCHAR3 FOR
:INT1 ROWS;
```

---

When you execute multiple INSERT statements in a batch, and the data source supports multi-row INSERT, the IBM Data Server Driver for JDBC and SQLJ uses multi-row INSERT to insert the rows. Multi-row INSERT can provide better performance than individual INSERT statements.

Example 5-30 shows a Java application program using `addBatch()`.

*Example 5-30 Using `addBatch()` in a Java program*

---

```
int[] testValuesInt = new int[] {1, 2, 3, 4};
String[] testValuesString = new String[] {"row1", "row2", "row3", "row4"};
PreparedStatement pstmt1 = con.prepareStatement ("INSERT INTO " +
tableName1 + " VALUES "
+ " (?,?) ");
for (int i = 0; i < 4; i++) {
    pstmt1.setInt (1, testValuesInt[i]);
    pstmt1.setString (2, testValuesString[i]);
    pstmt1.addBatch ();
}
pstmt1.executeBatch ();
```

---

## 5.7.6 Multi-row MERGE

The SQL MERGE statement introduced in DB2 9 for z/OS allows rows to be inserted into a table if they did not exist and updated when they do exist. It is also possible to specify the FOR N ROWS syntax to merge multiple rows.

For details on the MERGE statement, see the *DB2 Version 9.1 for z/OS SQL Reference*, SC18-9854.

Example 5-31 is an example of a MERGE statement in a Java application.

*Example 5-31 Multi-row MERGE in a Java program*

---

```
String mergeStmt = "MERGE INTO CMF001.T1 USING " +
    " (VALUES (?, ?, ?, ?)) AS TMP1 (C1, C2, C3, C4) ON TMP1.C1 = AGPK " +
    "WHEN NOT MATCHED THEN INSERT (C1, C2, C3, C4) " +
    "VALUES (TMP1.C1, TMP1.C2, TMP1.C3, TMP1.C4)";

java.sql.Statement s1 = c.createStatement ();

s1.executeUpdate (mergeStmt);
```

---

## 5.7.7 Heterogeneous batch updates

With `pureQuery`, you can batch INSERT, UPDATE, and DELETE statements that refer to different tables. These heterogeneous batch updates allow all associated tables to be updated in one network round trip to the server. With this means of heterogeneous batch updates, you call a method to indicate to `pureQuery` that you are starting a batch update.

Batch heterogeneous updates with parameters are described at the following Web page:

<http://publib.boulder.ibm.com/infocenter/idm/v2r1/topic/com.ibm.datatools.javatool.runtime.doc/topics/cpdqrunhetbthupdm1ttblwth.html>

## 5.7.8 Progressive streaming

Progressive streaming, also referred to as Dynamic Data Format, was introduced in the DB2 9 for z/OS server to optimize the retrieval of small and large LOB and XML data. It can also improve resource utilization on the DB2 for z/OS server in terms of freeing storage associated with LOB or XML data in a cursor-based scope (as opposed to a transaction-based scope). It is no longer necessary to use LOB locators. The server dynamically determines the mechanism to be used to return LOBs. Small LOBs (generally less than 12 K in size) may be inlined in the DRDA QRYDTA object. Larger LOBs (typically greater than 32 K in size) can be retrieved in chunks through progressive references (DRDA GETNXTCHK).

When using the JCC Type 4 driver, the following parameter values affect the use of progressive streaming:

- ▶ DB2BaseDataSource.progressiveStreaming = NOT\_SET (**default value**)
- ▶ DB2BaseDataSource.progressiveStreaming = YES
- ▶ DB2BaseDataSource.progressiveStreaming = NO

A value of YES (1) indicates that progressive streaming should be used if the data source supports it. A value of NO (2) indicates that progressive streaming should not be used. The Type 4 driver automatically uses a value of YES (1) when retrieving LOB or XML data that is not related to a stored procedure result set cursor.

No application changes are necessary to benefit from this performance enhancement and progressive references are handled under the covers by the drivers. Some customers were unable to disable the progressiveStreaming property and needed a server-side switch to disable this feature. APAR PK46079 adds the ability to disable Progressive Streaming if the following elements are present:

- ▶ The JCC property DB2BaseDataSource.progressiveStreaming = NOT\_SET
- ▶ DB2 for z/OS DSNZPARM PRGSTRIN=DISABLE

Refer to the IBM Data Server Driver for JDBC and SQLJ APAR IY99846 and DB2 for z/OS APAR PK46079 for details.

For a sample program that reads an input file and uses progressive references to retrieve chunks of XML data from the server, refer to Appendix C.3, “Progressive streaming” on page 439.

Table 5-5 on page 222 lists the Type 4 driver properties that are relevant when using dynamic data format.

*Table 5-5 Type 4 driver properties*

Driver Property	Description	Default
progressiveStreaming	Enables progressive streaming of LOB and XML data	Enabled by default for non-stored procedure result set cursors
streamBufferSize	Threshold that limits the size of LOB data that can be inlined	Generally 12 K

Driver Property	Description	Default
fullyMaterializeLOBData	Avoids the use of LOB locators	0 for servers that support dynamic data format

**Restriction:** Dynamic Data Format is not supported when DB2 9 for z/OS is acting as a DRDA Application Requester.

For more information, refer to *LOBs with DB2 for z/OS: Stronger and Faster*, SG24-7270.

## 5.7.9 SQL Interrupts

Support for SQL Interrupts was introduced in the DB2 for z/OS V8 server as a mechanism that allows long running SQL statements to be interrupted instead of cancelling the server thread and causing rollback. Because SQL interrupts do not interrupt threads held on locks, their applicability is not general, you may still need to cancel thread when the interrupt is issued.

You need to establish an additional remote connection from your application to the DB2 for z/OS server to issue the CLI /ODBC function `SQLCancel()` or invoke the JDBC cancel method. This is shown in Example 5-32.

When you cancel an SQL statement from a client application, you do not eliminate the original connection to the remote server. The original connection remains active to process additional SQL requests. Any cursor that is associated with the canceled statement is closed, and the DB2 server returns an `SQLCODE` of -952 to the client application when you cancel a statement by using this method. You can cancel only dynamic SQL codes that exclude transaction-level statements (`CONNECT`, `COMMIT`, `ROLLBACK`) from a client application.

**Tip:** You can use the `setQueryTimeout()` method to limit the number of seconds that SQL operations that use the given execution context object can execute. If an SQL operation exceeds the limit, an `SQLException` is thrown.

*Example 5-32 Issuing SQL interrupts*

```
public void run() {
    try {
        int i = 0;

        try {
            while (!parent.stmtStarted) {
                i++;
                Thread.sleep(20000); //Wait 20 seconds
                if (i > 2) {
                    break;
                }
            }
        } catch (Exception e) {
            ;
        }

        System.out.println("CancelThread: Issuing cancel");
        stmt.cancel();
    }
}
```

```

        System.out.println("CancelThread: Issued without any exception");
    } catch (SQLException e) {
        System.out.println("CancelThread Exception: " + getRealMessage(e));
        e.printStackTrace();
    }
    System.out.println("CancelThread Exit");
}

```

Some customers may experience application failures when migrating from a DB2 for z/OS V7 to a DB2 for z/OS V8 server when the applications have not been coded to tolerate the new error symptoms that could occur as a result of the SQL interrupt support. SQLINTRP was introduced by APAR PK41661 as a temporary measure to help customers with their migration process while they made the necessary application changes. The valid settings are ENABLE (the default) or DISABLE.

## 5.7.10 Remote external stored procedures and native SQL procedures

Table 5-6 can be used to compare and contrast external and native SQL procedures.

*Table 5-6 Native versus external procedures*

Remote external stored procedures	Remote native SQL procedures
Supported since DB2 Version 5	Introduced in DB2 9 for z/OS
Need to be parsed and translated to C	No need for generated C code and compilation.
Multiple versions of the procedure not supported.	Extensive support for Versioning.
Managed by WLM	Broken down into runtime structures like any SQL statement, not managed by WLM
SQL processing is run under a TCB, hence not zIIP-eligible.	Run under DBM1 enclave SRB, thus becoming zIIP eligible when called from a DRDA client.
Performs well when your stored procedure has a lot of application logic and uses math functions/string manipulation.	Performs well when your stored procedure is database-intensive and contains a lot of SQL but minimal logic.

Example 5-33 shows the creation and calling of a native SQL procedure using the Type 4 driver. Refer to Appendix C.1.1, “Using the Type 4 driver to call a native SQL procedure (BSQALone.Java)” on page 420 for the complete program.

*Example 5-33 Creating and calling a native SQL procedure using the Type 4 driver*

```

// Creating the native SQL procedure
stmt.executeUpdate("CREATE PROCEDURE BSQALONE ( IN VAR01 BINARY(5), "
+ "          IN VAR02 VARBINARY(5), "
+ "          INOUT VAR03 BINARY(5), "
+ "          INOUT VAR04 VARBINARY(5), "
+ "          OUT VAR05 BINARY(5), "
+ "          OUT VAR06 VARBINARY(5) )"
+ "VERSION VERSION1 "
+ "ISOLATION LEVEL CS "
+ "RESULT SETS 1 "
+ "LANGUAGE SQL "
+ " P1: BEGIN "

```

```
+ " DECLARE cursor1 CURSOR FOR "
+ "SELECT COLUMN4, COLUMN5 "
+ " FROM CDS_1 "
+ " WHERE COLUMN5 LIKE VAR03 AND COLUMN4 NOT LIKE VAR04; "
+ " SET VAR05 = VAR03; "
+ " SET VAR06 = VAR04; "
+ " OPEN cursor1; "
+ " END P1");
```

//<Several lines removed>

// Calling the native SQL procedure

```
cstmt = con
.prepareCall("{CALL PAOLR3.BSQL_ALONE(?,?,?,?)}");
```

```
byte[] inputByteArray1 = { 1,1,1};
cstmt.setBytes(1, inputByteArray1);
```

```
byte[] inputByteArray2 = { 2,2,2};
cstmt.setBytes(2, inputByteArray2);
```

```
byte[] inputByteArray3 = { 3,3,3};
cstmt.setBytes(3, inputByteArray3);
```

```
byte[] inputByteArray4 = { 4,4,4};
cstmt.setBytes(4, inputByteArray4);
```

```
cstmt.registerOutParameter(3, Types.BINARY); // IO
cstmt.registerOutParameter(4, Types.VARBINARY);
cstmt.registerOutParameter(5, Types.BINARY); // out as null
cstmt.registerOutParameter(6, Types.VARBINARY); // out as null
```

```
cstmt.execute();
```

---

Refer to the following resources for information:

- ▶ Using the IBM Data Studio Developer to create, test and deploy native SQL procedure  
[http://publib.boulder.ibm.com/infocenter/dstudio/v1r1m0/index.jsp?topic=/com.ibm.datatools.dwb.tutorial.doc/topics/dwb\\_abstract.html](http://publib.boulder.ibm.com/infocenter/dstudio/v1r1m0/index.jsp?topic=/com.ibm.datatools.dwb.tutorial.doc/topics/dwb_abstract.html)
- ▶ Debugging stored procedures on DB2 for z/OS using Data Studio (Part 1)  
[http://www.ibm.com/developerworks/data/library/techarticle/dm-0811zhang/?S\\_TACT=105AGX11&S\\_CMP=ART](http://www.ibm.com/developerworks/data/library/techarticle/dm-0811zhang/?S_TACT=105AGX11&S_CMP=ART)
- ▶ *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083

## 5.8 XA transactions

When using the DB2 for z/OS requester to connect to a DB2 for z/OS server, two-phase commit (2PC) is used by default. However, when the various data server drivers are used as requesters, the default commit protocol is one-phase commit (1PC) and you need to use XA transactions to use 2PC.

A global (XA) transaction is controlled and coordinated by an external transaction manager (external coordinator) to a resource manager. The transaction normally requires coordination across multiple resource managers that may reside on different platforms. To access an enterprise information system, the external coordinator sends an XID, which is defined by the X/Open XA standard, to a resource adapter. In addition to the length and FormatID fields, an XID has two other parts: the global transaction identifier (GTRID) and the branch qualifier (BQUAL).

### 5.8.1 Using the Type 4 driver to enable direct XA transactions

You would first need to create and register a data source and obtain an XA connection similar to what is shown in Example 5-34. In most scenarios, you will be using an Application Server such as WebSphere Application Server to develop your applications, so you would not have a need to use the explicit XA APIs as shown in Example 5-34.

If you are not using WebSphere, you need to download `jni.jar`, `providerutil.jar`, and `fscontext.jar` from `java.sun.com` and add them to your `CLASSPATH`. You first need to create an XA datasource and register it with JNDI (Java Naming and Directory Interface). JNDI is a Java API that provides a standard way to access objects in a registry. The most basic implementation uses a file-based system lookup. For the complete programs to create and register the XA datasource and enable two-phase commit, see Appendix C.2, "XA transaction samples" on page 423.

*Example 5-34 Enabling XA transaction through explicit XA API*

---

```

xaDS = (XADataSource)context.lookup (xau.getDataSourceName(serverType,
driverType,1));

    xaconn = xau.getXAConnection(xaDS);
    System.out.println ("XA Connection Obtained Successfully.....");

    conn = xaconn.getConnection();
    System.out.println ("Underlying Physical Connection conn: " +
conn.getClass().getName());
    //cleanup(conn);

    // Get the XA Resources
    xares = xaconn.getXAResource();

    Xid xid = xau.createRandomXid();
    Statement stmt1 = conn.createStatement();
    xares.start (xid, XAResource.TMNOFLAGS);
    xau.addToXidList(xid);
    int count1 = stmt.executeUpdate ("INSERT INTO H_POSUPDATE "
+ "VALUES (330,340)");

    xares.end(xid, XAResource.TMSUCCESS);

    System.out.println( "call xares.rollback(xid)" );
    xares.rollback(xid);
    System.out.println( "Closing XA connection" );
    xaconn.close();
    xaconn = null;
}

```

---

## 5.8.2 Using the IBM non-Java-based Data Server Drivers/Clients to enable direct XA transactions

Starting with Version 9.5 FixPack 3, IBM Data Server Clients and non-Java-based Data Server Drivers that have a DB2 Connect license can directly access a DB2 for z/OS Sysplex and use native XA support without going through a middle-tier DB2 Connect server.

This type of client-side XA support is only available for transaction managers that use a single-transport processing model. In a single-transport model, a transaction, over a single transport (physical connection), is tied to a member from `xa_start` to `xa_end`. The transaction end is followed immediately by `xa_prepare(readonly)`, `xa_prepare` plus `xa_commit` or `xa_rollback`, or `xa_rollback`. All of this must occur within a single application process. Examples of transaction managers that use this model include IBM TXSeries® CICS®, IBM WebSphere® Application Server, and Microsoft® Distributed Transaction Coordinator. Enable XA support by using the `SINGLE_PROCESS` parameter in the `xa_open` string, or by specifying `enableDirectXA = true` in the `db2dsdriver` configuration file.

**Important:** DB2 for z/OS (V8 and 9) APAR PK69659 must be installed for direct XA support (needed for transaction managers such as Microsoft Distributed Transaction Coordinator).

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.dbconn.doc/doc/t0054754.html>

The ADO .NET provider for DB2 provides distributed transaction support. The sample provided is a windows form application written in Visual Basic. The sample demonstrates the distributed transaction support provided by the ADO .NET provider for DB2. The application opens two connections to the DB2 subsystem on z/OS. The application then inserts rows into two tables, one from each connection, to demonstrate distributed transaction.

**Tip:** When using the non-Java-based Data Server Client or Runtime Client, you need to set `enableDirectXA=true` in the `db2dsdriver.cfg` file or specify `SINGLE_PROCESS` in the CLI script. When using the non-Java-based data server drivers, this support is implicitly provided and is the default behavior.

Figure 5-12 indicates that the Enable XA Transactions box must be checked before attempting XA connections using the .NET driver.

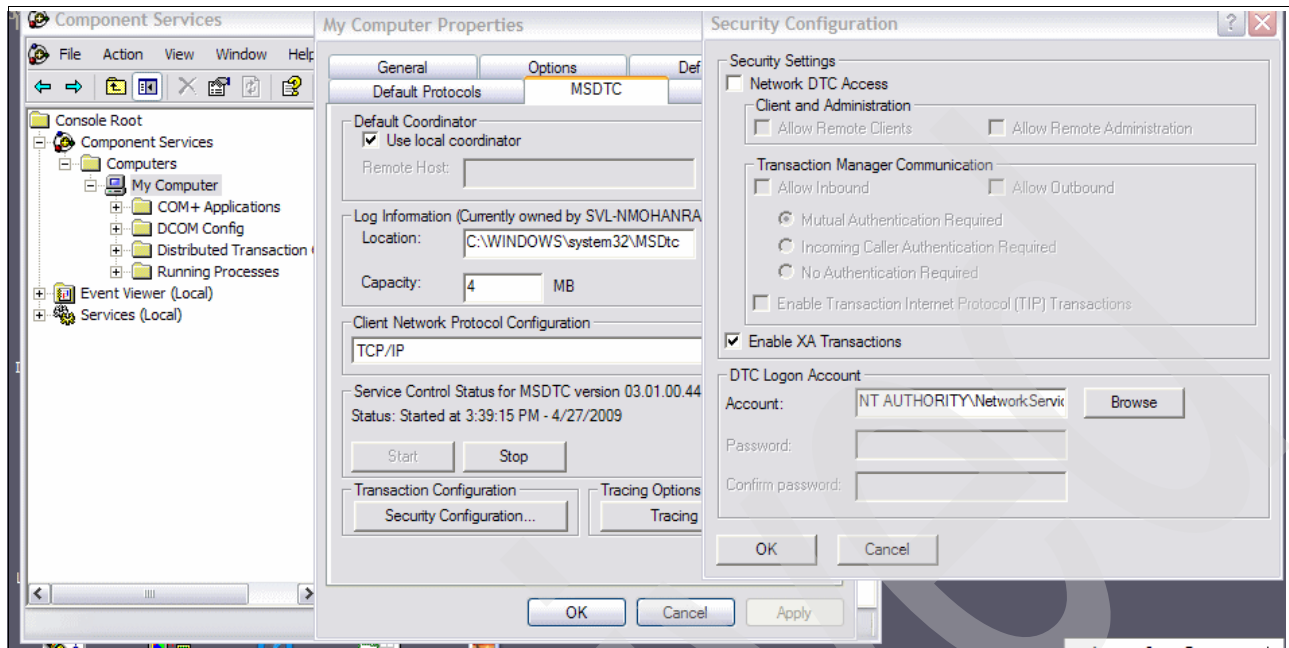


Figure 5-12 Enabling XA transaction



Figure 5-13 shows a sample application developed using .NET. The associated db2dsdriver.cfg file has been shown in Example 5-16 on page 203.

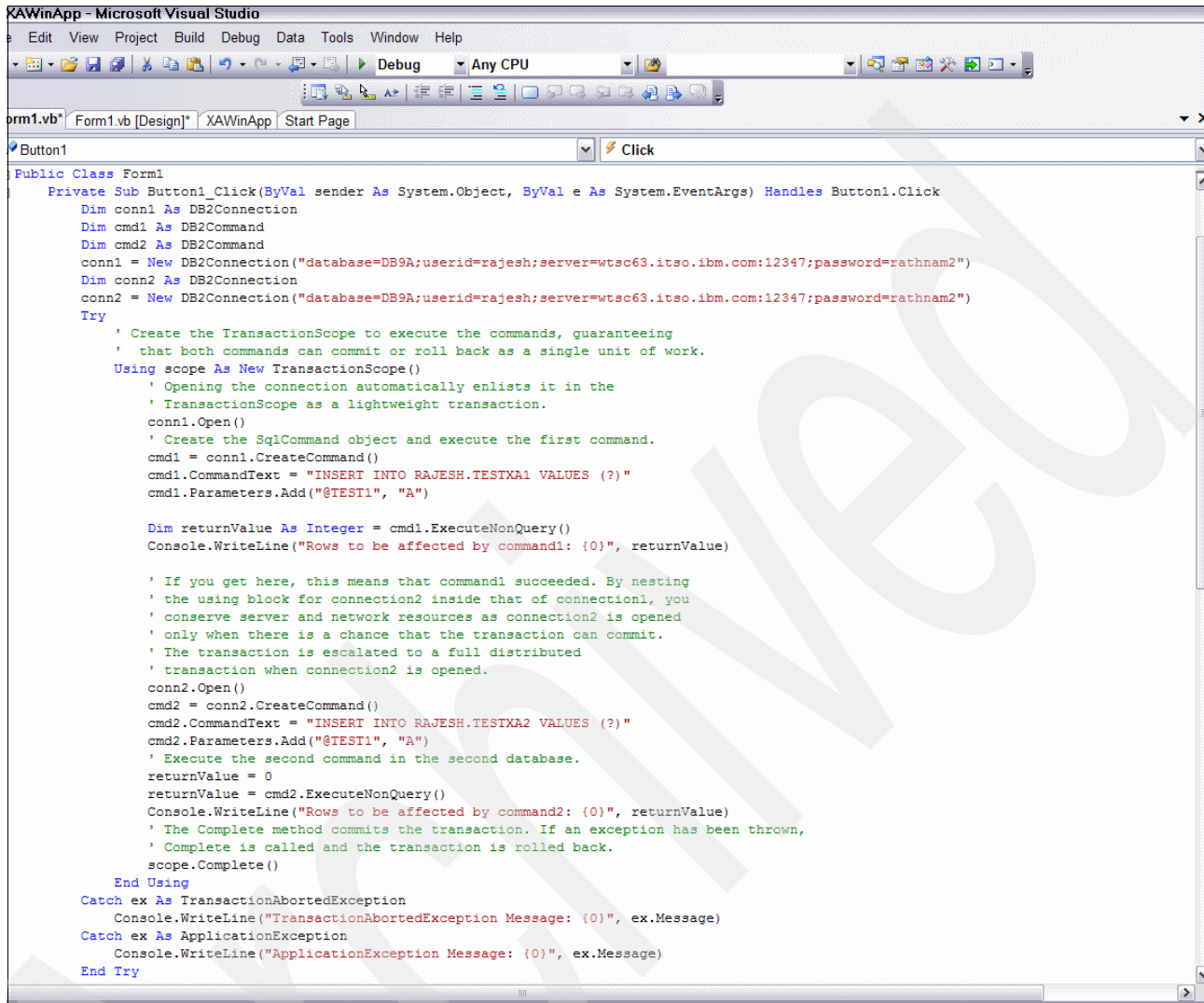


Figure 5-13 .NET application that uses XA transactions

## 5.9 Remote application recommendations

Here is a quick summary of recommendations when developing distributed applications:

- ▶ Use the WHERE, GROUP BY, and HAVING clauses to limit the size of your result set.
- ▶ Use OPTIMIZE for n ROWS and FETCH FIRST n ROWS whenever possible to get both access path selection and DRDA blocking benefits including extra blocks.
- ▶ Declare your cursor with FOR FETCH ONLY, or FOR READ ONLY, and INSENSITIVE STATIC to allow the server to use block fetch.
- ▶ Use CURRENTDATA(NO) and ISOLATION(CS) when possible. Avoid ISOLATION(RR).
- ▶ Use stored procedure result sets.
- ▶ Use COMMIT on RETURN clause for stored procedures that do not return result sets.

- Use dynamic data format for LOBs, multi-row fetch and multi-row insert where supported by the drivers.
- COMMIT on business transaction boundaries but do not use autocommit (default for CLI applications).
- Avoid using WITH HOLD cursor.
- Free resources you no longer need, that is, explicitly close cursors after you have fetched all data, declare DGTs with ON COMMIT DROP TABLE, free LOB locators.
- Use KEEP DYNAMIC(YES) for applications that use very few SQL statements very frequently to avoid excessive prepares and keep in mind that it prevents the connection from being inactivated.
- Use connection pooling and connection concentration to speed up connection processing.

Table 5-7 compares and contrasts the various requesters.

Table 5-7 Comparison across requesters

	Ease of installation/size of footprint	Ease of coding applications (GUI tools like Data Studio Developer)	Performance	Support for Sysplex WLB, seamless failover/ACR (data sharing)
<b>DB2 9 for z/OS Requester</b>	Most complex installation	Use when you have existing applications.	Supports both static and dynamic SQL.	Not supported
<b>Type 4 driver</b>	Easy to install /small footprint	For existing Java-based dynamic applications.	Dynamic SQL only, Performs well when cache hit ratio is high.	Supported
<b>SQLJ</b>	Easy to install	For existing Java-based static applications.	Supports both static and dynamic SQL.	Supported
<b>pureQuery using Type 4 driver</b>	Easy to install	Easiest to code. Recommended for NEW Java-based static applications.	Supports both static and dynamic SQL.	Supported
<b>Data server drivers in ODBC/CLI environment</b>	Easy to install/small footprint.	For C/C++ applications.	Dynamic SQL only, performs well when cache hit ratio is high.	Supported directly as well as through DB2 Connect server
<b>Data server drivers in .NET environment</b>	Easy to install/small footprint	For C and VisualBasic applications	Dynamic SQL only, performs well when cache hit ratio is high.	Supported

Table 5-8 lists the fetch/insert features supported by the DB2 9 for z/OS requester and various client drivers against the DB2 9 for z/OS server.

Table 5-8 Fetch/insert feature support by client/driver

Requester/Feature	Limited block FETCH	Progressive Streaming for LOB and XML data (Dynamic Data Format)	Multi-row FETCH	Multi-row INSERT
<b>DB2 9 for z/OS requester</b>	Supported for blocking cursors	Not supported	Supported when explicit rowset syntax is used or implicitly when using DSNTEP4	Supported when explicit FOR MULTIPLE ROWS syntax is used
<b>JCC Type 4 driver/pureQuery</b>	Supported for blocking cursors	Supported by default for both LOB and XML data. Can be disabled.	Supported by default for scrollable cursors.	Heterogeneous batch updates (INSERT/MERGE) supported through pureQuery APIs.
<b>Data server driver in ODBC/CLI environment</b>	Supported for blocking cursors.	Supported by default only for LOB data. Can be disabled	Supported through SQLBulkOperations( )	Supported through array input chaining
<b>Data server driver in .NET environment</b>	Supported for blocking cursors.	Supported by default only for LOB data. Cannot be disabled	Not supported	Supported through array input chaining and DB2BulkCopy

Archived

## Data sharing

In this chapter we discuss considerations on setting up a data sharing group in a distributed environment where availability, workload balancing, and failover management are the primary objectives.

This chapter contains the following sections:

- ▶ “High availability aspects of DRDA access to DB2 for z/OS” on page 234
- ▶ “Recommendations for common deployment scenarios” on page 247
- ▶ “DB2 failover scenario with and without Sysplex Distributor” on page 259
- ▶ “Migration and coexistence” on page 265

## 6.1 High availability aspects of DRDA access to DB2 for z/OS

In this section we focus on DRDA access to a DB2 data sharing group to ensure that remote applications get the highest possible level of availability accessing DB2 for z/OS.

The principal vehicle for high availability of DB2 for z/OS is the exploitation of System z parallel sysplex technology. Consequently, we focus on continuous availability of a DB2 for z/OS server by exploring the seamless integration of DRDA connections with a DB2 data sharing group.

We use DB2 Connect V9.5 FP3 or later and equivalent DRDA AR clients for distributed platforms supporting sysplex workload balancing (sysplex WLB). Supported DRDA AR clients include: DB2 Connect server/client, Type 4 driver, and the non-Java-based IBM Data Server Driver (CLI Driver and .NET Provider).

The test for deployment verification was done using the trade workload described in Appendix B, “Configurations and workload” on page 401.

**Note:** DB2 for z/OS Requester does not have same level of capability for transaction pooling, but it does provides connection level load balancing.

### 6.1.1 Key components for DRDA high availability

To understand DRDA integration with the high availability capability of DB2 for z/OS, it is important to know the functions of some key components of a parallel sysplex and of a DB2 for z/OS data sharing group:

- ▶ z/OS Workload Manager (WLM)

WLM provides a server list to the DRDA AR. The server list indicates the available DB2 for z/OS data sharing members and their relative weights.

- ▶ DB2 Connect server/client, the Type 4 driver, and the non-Java-based IBM Data Server Driver

These products include sysplex awareness capability that provides transparent transaction routing based on the weights of the available members of the DB2 for z/OS data sharing group.

- ▶ TCP/IP stack

- Virtual IP Addressing (VIPA) isolates the physical network path from IP addressing.
- Dynamic VIPA (DVIPA) provides network resiliency against TCP/IP address space or z/OS outages.
- Distributed DVIPA offers the capability to connect to a single DVIPA to distribute service provided by several stacks, providing isolation from an outage of a DB2 for z/OS member.

### 6.1.2 z/OS WLM in DRDA workload balancing

WLM plays important roles in terms of using sysplex workload balancing (WLB). The sysplex WLB is based on a DRDA “server list”, which consists of lists of IP address for DB2 data sharing member, port number, and “WLM server weight” (weight). The server list is created and maintained by WLM, and will be passed to DRDA AR client through DRDA flow during

connect processing. When transaction pooling is used, DRDA AR clients will use the updated server list on each transaction to distribute workload to DB2 data sharing members. DRDA AR clients get the server list when heavy reuse is used.

There are no configuration parameters related to enabling sysplex WLB on DB2 for z/OS server. When DDF is started, it registers itself to WLM, unless MAXDBAT is set to 0 or DDF is stopped. In either of these exception cases, that member's DDF does not appear in the server list. If you stop DDF, DDF de-register's from WLM and the member is removed from the server list.

There are several factors WLM will take into account when creating and updating weights. These factors follow:

- ▶ Displaceable capacity of systems (base information)
- ▶ Enclave service class achievement (Performance Index, or PI)
 

WLM goal should be attainable when system is not under stress, which would result in a PI < 1. If a service class has a PI > 1, it may affect workload balancing (as of z/OS V1R7 or higher, and PK03045 for DB2 for z/OS V8).
- ▶ Enclave service class request queuing
- ▶ Availability of general CP and zIIP (z/OS V1R9 or higher)
 

The weight of server list will reflect a combination of general purpose processor and zIIP for available capacity.
- ▶ DB2 9 for z/OS Health
 

DB2 will report its health factor of 0 to 100 to WLM based on the current storage consumption within the DBM1 and DIST address spaces.

**Note:** zIIP awareness was introduced in APAR PK38867 for DB2 V8 and 9. Prior to z/OS V1R9, WLM sysplex routing service returned only weight of a server based on available capacity of general processors adjusted by the health of the server and queuing at the server. With this change, z/OS V1R9 WLM will return the weight of the available capacity of a server with general CP, zIIP, and zAAP processors. The combined weight depends on all servers running z/OS V1R9 or later.

You need to know how your WLM policies are defined for your DRDA workload and how your DRDA workloads meet the goal you defined, using RMF reports and server list informations.

Example 6-1 shows an RMF workload activity report from our test. You should have good number in PI(<1) when system is not under stress. Having PI > 1 could mean that sysplex WLB be affected.

Example 6-1 RMF workload activity report example

1

WORKLOAD ACTIVITY

PAGE 18

z/OS V1R10

SYSPLEX SANDBOX

DATE 04/23/2009

INTERVAL 09.59.748

MODE = GOAL

RPT VERSION V1R10 RMF

TIME 17.39.37

POLICY ACTIVATION DATE/TIME 04/22/2009 20.38.49

----- SERVICE CLASS PERIODS

REPORT BY: POLICY=OVER

WORKLOAD=DATABASE

SERVICE CLASS=DDFTST  
CRITICAL =NONE

RESOURCE GROUP=\*NONE

PERIOD=1 IMPORTANCE=2

-TRANSACTIONS-

TRANS-TIME HHH.MM.SS.TTT

--DASD I/O--

---SERVICE---

SERVICE TIME

---APPL %---

--PROMOTED--

----STORAGE----

AVG 0.00 ACTUAL 23 SSCHRT 0.0 IOC 0 CPU 0.004 CP 0.00 BLK 0.000 AVG 0.00

MPL 0.00 EXECUTION 23 RESP 0.2 CPU 124 SRB 0.000 AAPCP 0.00 ENQ 0.000 TOTAL 0.00

ENDED 1 QUEUED 0 CONN 0.1 MSO 0 RCT 0.000 IIPCP 0.00 CRM 0.000 SHARED 0.00

END/S 0.00 R/S AFFIN 0 DISC 0.0 SRB 0 IIT 0.000

GOAL: RESPONSE TIME 000.00.00.500 FOR 80%

	RESPONSE TIME	EX	PERF	AVG	--EXEC USING%--	----- EXEC DELAYS % -----						-USING%-	--- DELAY % ---				%
SYSTEM	ACTUAL%	VEL%	INDX	ADRSP	CPU	AAP	IIP	I/O	TOT		CRY	CNT	UNK	IDL	CRY	CNT	QUI
SC63	100	N/A	0.5	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0

You can monitor the weights of each DB2 data sharing member using the `DISPLAY DDF DETAIL` command. The command has been enhanced in APAR PK80474 for DB2 V8 and 9.

**Note:** APAR PK80474 adds several enhancements to DISPLAY DDF commands, including display server list, show Group IP address, member specific IP address, and defined location aliases. For DB2 for z/OS V8, the capability to display location alias, already supported in DB2 9, has also been added.

Example 6-2 displays the location server list of our 3-member DB2 data sharing group: D9C1 (9.12.4.103), D9C2 (9.12.4.104), and D9C3 (9.12.4.105). D9C2 has the smallest weight (18) because the other two members have each installed two zIIPs.

*Example 6-2 Example of the server list from the DISPLAY DDF DETAIL command*

```

-D9C1 DIS DDF DET
DSNL080I -D9C1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 335
DSNL081I STATUS=STARTD
DSNL082I LOCATION LUNAME GENERICLU
DSNL083I DB9C USIBMSC.SCPD9C1 -NONE
DSNL084I TCPDPORT=38320 SECPORT=0 RESPORT=38321 IPNAME=-NONE
DSNL085I IPADDR=:9.12.4.102
DSNL086I SQL DOMAIN=d9cg.itso.ibm.com
DSNL086I RESYNC DOMAIN=d9cg.itso.ibm.com
DSNL087I ALIAS PORT SECPORT
DSNL088I DB9CALIAS 38324 0
DSNL088I DB9CSUBSET 38325 0
DSNL089I MEMBER IPADDR=:9.12.4.103
DSNL090I DT=I CONDBAT= 300 MDBAT= 100
DSNL092I ADBAT= 8 QUEDBAT= 0 INADBAT= 0 CONQUED= 0
DSNL093I DSCDBAT= 3 INACONN= 19
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR IPADDR
DSNL102I 45 :9.12.4.105
DSNL102I 42 :9.12.4.103
DSNL102I 18 :9.12.4.104
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

If you are using the Type 4 driver to connect to DB2 for z/OS, you can get a trace from the global transport object pool to monitor connection concentrator and sysplex workload balancing as shown in Example 6-3 on page 238.



In this example, the counters are as follows:

- ▶ npr  
The total number of requests that the Type 4 driver has made to the pool since the pool was created.
- ▶ nsr  
The number of successful requests that the Type 4 driver has made to the pool since the pool was created. A successful request means that the pool returned an object.
- ▶ lwroc  
The number of objects that were reused but were not in the pool. This can happen if a Connection object releases a transport object at a transaction boundary. If the Connection object needs a transport object later, and the original transport object has not been used by any other Connection object, the Connection object can use that transport object.
- ▶ hwroc  
The number of objects that were reused from the pool.
- ▶ coc  
The number of objects that the Type 4 driver created since the pool was created.
- ▶ aocc  
The number of objects that exceeded the idle time that was specified by `db2.jcc.maxTransportObjectIdleTime` and were deleted from the pool.
- ▶ rmoc  
The number of objects that have been deleted from the pool since the pool was created.
- ▶ nbr  
The number of requests that the Type 4 driver made to the pool that the pool blocked because the pool reached its maximum capacity. A blocked request might be successful if an object is returned to the pool before the `db2.jcc.maxTransportObjectWaitTime` is exceeded and an exception is thrown.
- ▶ tbt  
The total time in milliseconds for requests that were blocked by the pool. This time can be much larger than the elapsed execution time of the application if the application uses multiple threads.
- ▶ sbt  
The shortest time in milliseconds that a thread waited to get a transport object from the pool. If the time is under one millisecond, the value in this field is zero.
- ▶ lbt  
The longest time in milliseconds that a thread waited to get a transport object from the pool.
- ▶ abt  
The average amount of time in milliseconds that threads waited to get a transport object from the pool. This value is `tbt/nbr`.
- ▶ tpo  
The number of objects that are currently in the pool.

*Example 6-3 Example trace output from the global transport objects pool*

---

```
2009-04-29-16:14:46.111 ageServerLists, 8000 2000 4
2009-04-29-16:14:54.113 ageServerLists, 8000 2000 4
2009-04-29-16:14:55.712| Scheduled PoolStatistics npr:6347 nsr:6353 lwroc:6294
hwroc:0 coc:49 aooc:0 rmoc:0 crr:0 nbr:0 nbds:0 nbpm:0 sbt:0 abt:0 lbt:0 tbt:0
tpo:49
2009-04-29-16:15:02.121 ageServerLists, 8000 2000 4
2009-04-29-16:15:10.131 ageServerLists, 8000 2000 4
2009-04-29-16:15:24.261 ageServerLists, 8000 2000 4
2009-04-29-16:15:32.360 ageServerLists, 8000 2000 4
2009-04-29-16:15:40.369 ageServerLists, 8000 2000 4
2009-04-29-16:15:48.436 ageServerLists, 8000 2000 4
2009-04-29-16:15:55.716| Scheduled PoolStatistics npr:9721 nsr:9727 lwroc:9655
hwroc:0 coc:52 aooc:1 rmoc:1 crr:0 nbr:0 nbds:0 nbpm:0 sbt:0 abt:0 lbt:0 tbt:0
tpo:51
2009-04-29-16:15:56.440 ageServerLists, 8000 2000 4
2009-04-29-16:16:04.448 ageServerLists, 8000 2000 4
2009-04-29-16:16:12.454 ageServerLists, 8000 2000 4
2009-04-29-16:16:20.621 ageServerLists, 8000 2000 4
2009-04-29-16:16:28.699 ageServerLists, 8000 2000 4
2009-04-29-16:16:36.709 ageServerLists, 8000 2000 4
2009-04-29-16:16:44.715 ageServerLists, 8000 2000 4
2009-04-29-16:16:52.721 ageServerLists, 8000 2000 4
2009-04-29-16:16:55.717| Scheduled PoolStatistics npr:11749 nsr:11755 lwroc:11681
hwroc:0 coc:52 aooc:2 rmoc:2 crr:0 nbr:0 nbds:0 nbpm:0 sbt:0 abt:0 lbt:0 tbt:0
tpo:50
2009-04-29-16:17:00.724 ageServerLists, 8000 2000 4
```

---

If you are using DB2 Connect server, you can also display your server list by using the db2pd command with -sysplex option. You can also see the number of connections made to a specific member. See Example 6-4.

*Example 6-4 Display server list information from DB2 Connect Server*

---

```
$ db2pd -sysplex
```

```
Database Partition 0 -- Active -- Up 0 days 00:06:06
```

```
Sysplex List:
```

```
Alias: DB9C
```

```
Location Name: DB9C
```

```
Count: 3
```

IP Address	Port	Priority	Connections	Status	PRDID
9.12.4.105	38320	53	1	0	DSN09015
9.12.4.103	38320	48	0	0	
9.12.4.104	38320	21	0	0	

---

**Tip:** Use one or a combination of these reports together with the RMF workload activity report to make sure the system is not under stress.

### 6.1.3 The sysplex awareness of clients and drivers

DB2's primary approach for scalability and high availability is to exploit the clustering capabilities of the System z Parallel Sysplex. In a DB2 data sharing environment, multiple instances of DB2 data sharing member can all access the same databases. If one member fails or cannot be reached, the workload can be dynamically re-routed to other members of the data sharing group, continuing to work.

DRDA connections can exploit the sysplex as follows:

- ▶ DRDA AR can connect applications to a DB2 data sharing group as though it were a single database server, and spread the workload among the different members, based on server lists dynamically provided by WLM.
- ▶ DRDA AR can recognize when a member of a DB2 data sharing group fails and can automatically route new connections to other members.

Initially, the sysplex awareness capability (sysplex WLB) was provided by the DB2 Connect server (known as Gateway). Today, the T4 Driver and the non-Java-based IBM Data Server Driver and Client products have been enhanced to provide sysplex WLB.

**Note:** The sysplex WLB of CLI Diver and .NET Provider was introduced in DB2 Connect V9.5 FP3 or later and any equivalent level of the non-Java-based IBM Data Server Driver or Client.

Figure 6-1 on page 240 illustrates how DRDA AR clients on distributed platforms provided by IBM can be configured to connect to the DB2 data sharing group. A client can connect to any of the members. From the point of view of the application on the remote platform, the DB2 data sharing group, DB9C, appears as a single database server, as shown in Figure 6-1 on page 240.

**Note:** To take advantage of sysplex WLB with one of the DRDA clients, you need to have connection pooling capability where remote applications can reuse physical connections between DB2 and applications.

DB2 V9.5 FP4 introduced sysplex WLB without connection pooling

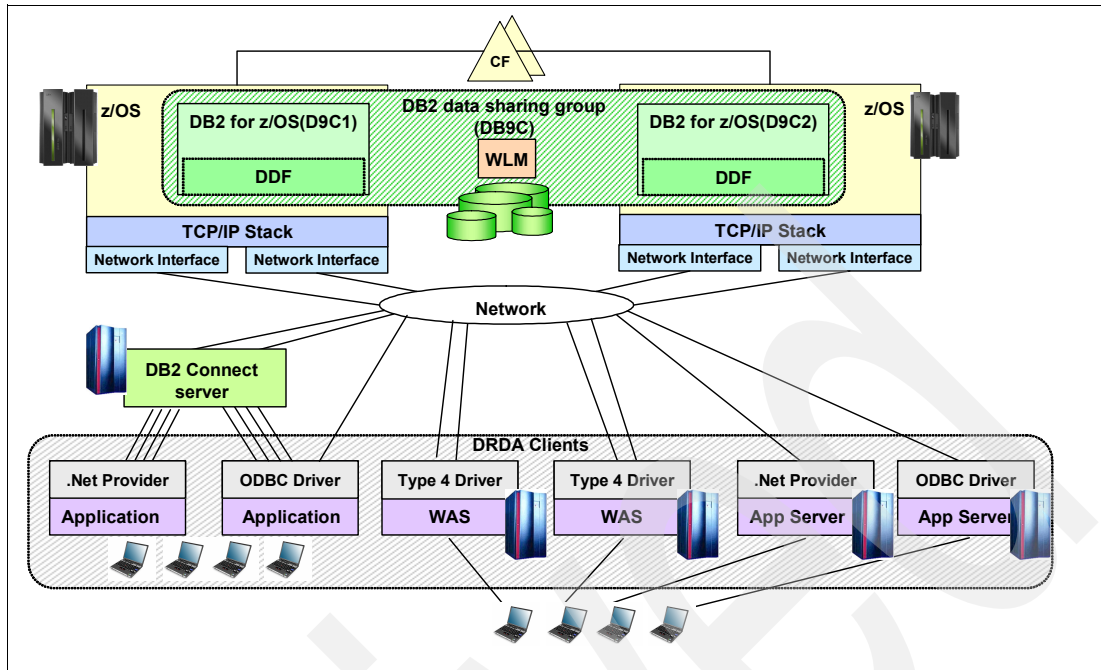


Figure 6-1 DRDA access to DB2 data sharing

For the DRDA AR clients, connections to a DB2 data sharing group are established exactly the same way as connections to a non-data sharing DB2 subsystem. The information for the chosen DB2 data sharing member must be catalogued to the appropriate configuration. After an initial successful connect, the DRDA AR is automatically able to establish connections to either member using the server list.

The logic flow of DRDA AR clients for obtaining and using a server list is illustrated in Figure 6-2 on page 241. In this example, the DRDA AR client has catalogued the DB2 data sharing member as a single DB2 subsystem: DB9C at 9.12.4.102, which represents distributed DVIPA explained in 6.1.4, “Network resilience using Virtual IP Addressing” on page 242. The DRDA AR clients can be any one of following: the Type 4 driver, the non-Java-based IBM Data Server Driver or Client, or a DB2 Connect server.

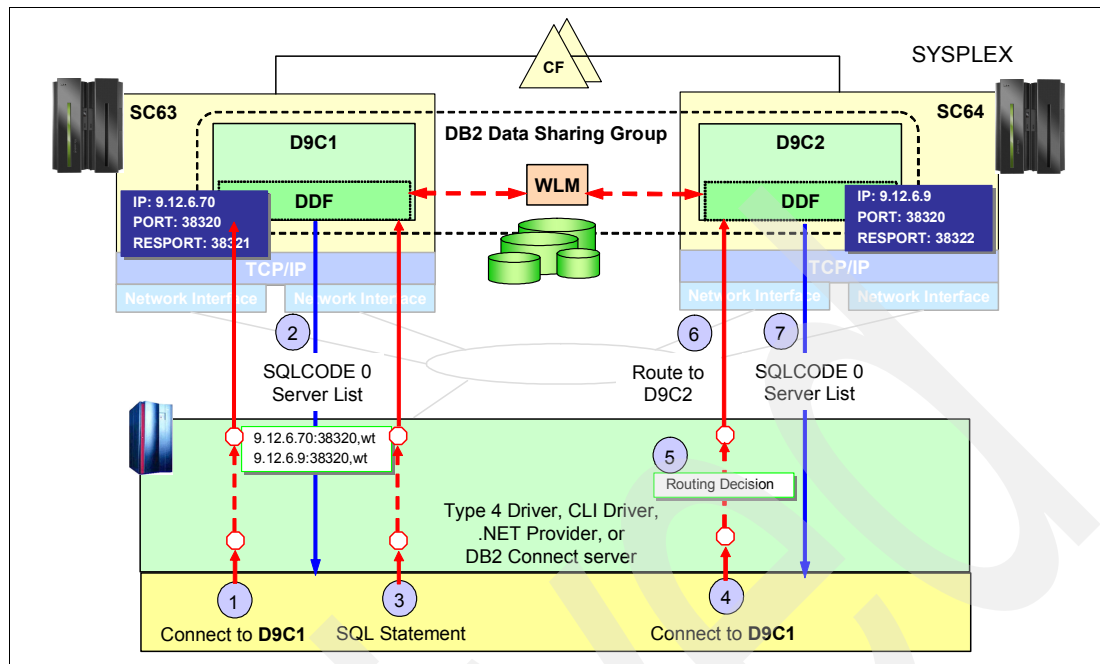


Figure 6-2 The logical flow of DRDA AR clients obtaining server information

The sequence of action in the diagram is as follows:

1. The first connect request is presented by the DRDA AR client. It is a connect to database DB9C, which is correctly cataloged at TCP/IP address 9.12.4.102, with sysplex WLB turned on, so that the DRDA AR client knows the target is a DB2 data sharing group. The DRDA AR client flows the connection to member D9C1, which was routed by Sysplex Distributor. As this is the first connection, the DRDA AR client is unaware of number of members available in data sharing environment, and Sysplex Distributor is responsible for routing the first connection to DB2 for z/OS.
2. Once the connection to DB9C is established, server lists, which comprise a list of available data sharing members and WLM information (weight) for the relative load of each member, will be returned to the DRDA AR client. The DRDA AR client caches this information for use with subsequent connections.
3. The DRDA AR client continues to route all SQL traffic for this first transaction to member D9C1, which was directed by Sysplex Distributor.
4. A second transaction request is initiated by the DRDA AR client. In the application, this request is also directed to database D9C1.
5. The DRDA AR client uses the server list information to decide to which member to route the second transaction. This decision is based on connectivity information and WLM information received by the first connect.
6. On this occasion, the DRDA AR client chooses to route this connection request to member D9C2.
7. After successful connection establishment (transparent to the application), the DRDA AR client receives updated server lists with member availability and WLM data for use during the next sysplex connection decision.

The example in Figure 6-2 shows the IP configuration at the beginning of our project.

**Restriction:** When you are using DB2 Connect server, connection concentrator needs to be configured to work as described above. Otherwise, server list are updated when each physical connection is made.

For all other DRDA AR clients, transaction pooling is automatically turned on when you turned on sysplex WLB.

### How sysplex workload balancing works

Take Table 6-1 as an example of a server list taken from our test to explain how sysplex WLB works. The weight will be calculated as a ratio. This means that about 42% of the workload will be routed it to D9C1, about 17% to D9C2, and about 42% to D9C3. (This example shows the member DVIPA addresses.)

Table 6-1 Server list and calculated ratio

Member	Weight	Ratio (weight/total weight)
D9C1(9.12.4.103)	53	0.41732284
D9C2(9.12.4.104)	21	0.16535433
D9C3(9.12.4.105)	53	0.41732284

Assume the Type 4 driver (or any other DRDA AR client that supports sysplex WLB) has assigned connections to the transport as per the numbers of active connections (connections hold transports) assigned by Type 4 driver, as shown in Table 6-2.

Table 6-2 The numbers of active connections

Member	Active connections	Ratio (active/total)
D9C1(9.12.4.103)	19	0.452381
D9C2(9.12.4.104)	7	0.166667
D9C3(9.12.4.105)	16	0.380952

When a new transaction arrives, the weight ratios are recalculated according to the new server list, and active connection ratios would be evaluated using them. The lowest weighted member would be checked first, then the higher weighted member. Assume, in this example, server list weight did not change. In case of Table 6-2, D9C3 will get the next new transaction since D9C1 and D9C2 have higher active transaction ratios than associated weight ratios.

**Note:** The described algorithm is based on DB2 Connect V9.5 FP3 or later and equivalent level of the T4 Driver and non-Java-based IBM Data Server Driver or Client with sysplexWLB enabled.

## 6.1.4 Network resilience using Virtual IP Addressing

Traditionally, an IP address is associated with each physical link and is unique across the entire visible network. Within those IP routing network, failure of any intermediate link or adapter means failure of application network service unless they are an alternate path of routing the network. The router can route network traffic around failed intermediate links but if the link associated to the destination fails, there is no way for IP routing network to provide an alternate path to application. The virtual IP address (VIPA) of z/OS Communication Server

removed this limitation by disassociating the IP address from a physical link and associating it with the TCP/IP address space or a stack. This type of VIPA is called static VIPA. Because a stack is just another address space in z/OS, the failure of the physical interface can be extended to the failure or planned outage of a stack, or failure of the entire z/OS. Moving VIPA across the stacks is needed in case of stack outage; this process is called VIPA Takeover.

Dynamic VIPA (DVIPA) automates VIPA Takeover to a backup stack. Distributed DVIPA enables z/OS Sysplex Distributor improved function by allowing a request to a single DVIPA to be served by applications on several stacks listed in the configuration. This adds the benefit of limiting application exposure to stack failure, while providing additional benefit of connection level work load balancing.

- **Static VIPA**

A static VIPA is an z/OS TCP/IP facility to provide alternate network routing to the same DB2 for z/OS subsystem or member in the event of a physical network failure. A static VIPA cannot reroute connections to a different DB2 for z/OS system.

- **DVIPA**

Provides network resilience by providing VIPA Takeover to a backup stack.

- **Distributed DVIPA**

Represented DVIPA of z/OS Sysplex Distributor, which provides application resilience by rerouting the DRDA TCP/IP traffic to a different member of a DB2 for z/OS data sharing group. DVIPAs should be used in conjunction with the z/OS Sysplex Distributor.

In non-data sharing DRDA connection environment, Static VIPA or DVIPA is recommended for network resilience. In case of network interface outage, existing connections are automatically rerouted to the same DB2 for z/OS subsystem, transparent to the connected applications (assuming an alternate network route exists).

Distributed DVIPA and sysplex WLB provided by DRDA AR clients is recommended for DRDA connections to DB2 for z/OS data sharing environment.

### **6.1.5 Advanced high availability for DB2 for z/OS data sharing**

The combination of distributed DVIPA, or z/OS Sysplex Distributor, and client capabilities provides applications with further levels of resilience.

- **Advanced Member routing.** DRDA AR clients rely on the server lists of DB2 data sharing group returned from server to distribute workload across DB2 data sharing group. Without Sysplex Distributor, the first connection from an application relies on the availability of the catalogued DB2 data sharing member. Sysplex Distributor provides the capability to connect to any available DB2 data sharing member without re-cataloging in DRDA AR clients.
- **Automatic client reroute** to a different member of a data sharing group, in the event of the failure of one member.

#### **Distributed DVIPA and Sysplex Distributor**

Figure 6-3 on page 244 illustrates the concept of DRDA connections to the DB2 data sharing group through Sysplex Distributor. The DRDA AR client system simply catalogs the DB2 data sharing group at the Dynamic VIPA of the Sysplex Distributor (9.12.4.102) as though you are connecting to a single DB2 server. Once a new connection request is made, it is routed to the most appropriate member, based on availability and WLM information delivered from DB2 for z/OS to DRDA AR clients.

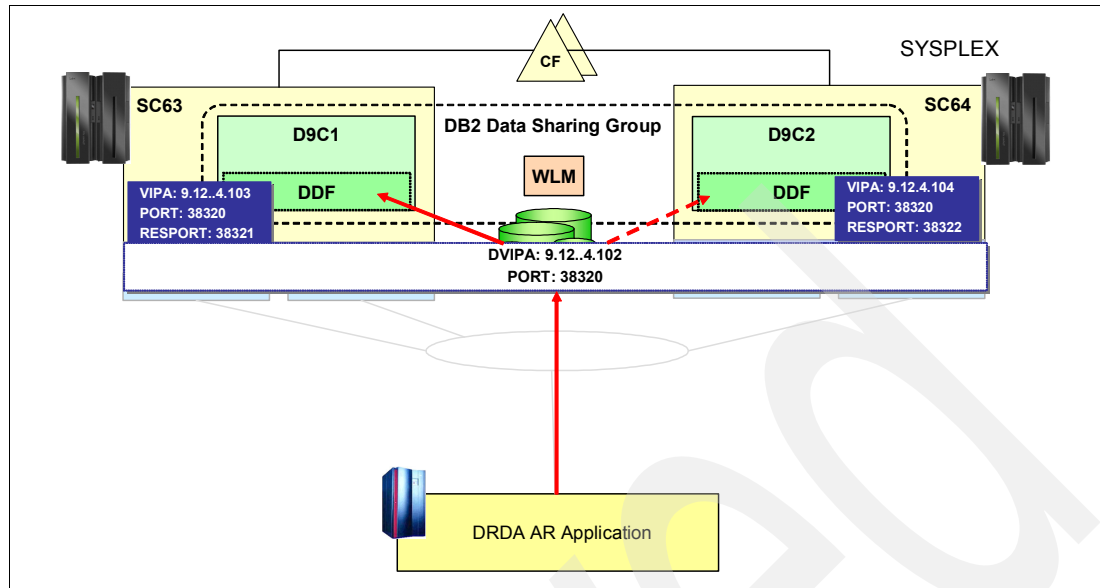


Figure 6-3 Sysplex Distributor connection assignment

The specification of enabling sysplex WLB in the Type 4 driver or the non-Java-based IBM Data Server Driver or Client by parameter (example will be shown in the following sections), or in DB2 Connect Server by specifying the SYSPLEX parameter on the DCS database directory entry is essential to ensure that the WLM workload distribution is managed by DRDA AR clients, based on DB2 WLM information. Sysplex Distributor only performs network load balancing for initial connection. After an initial connection is made and a server list is sent to DRDA AR clients, DRDA AR clients will use DB2 data sharing member specific IP addresses to connect to appropriate DB2 data sharing members.

The following explanation gives Sysplex Distributor-related definitions on TCP/IP stack. When you define Sysplex Distributor, you need to define two different roles for each of the TCP/IP stacks participating in the Sysplex Distributor. One is the distribution stack or distributed DVIPA, which represent the Sysplex Distributor. The other one is the target stack, which provides the service, like DB2 for z/OS. You can give both roles to one stack but you will need to define a backup for the distribution stack. In Figure 6-3, one of two participating TCP/IP stacks, represented on SC63 and SC64, will become the primary distribution stack and the other will become the backup stack. Both stacks take on the role of target stack.

Set following statements in the IPCONFIG statement in all your participating stacks:

- ▶ DATAGRamfwd  
Enables the transfer of data between networks
- ▶ DYNAMICXCF  
Indicates that XCF dynamic support is enabled
- ▶ SYSPLEXRouting  
Specifies that this TCP/IP host is part of an sysplex domain and should communicate interface changes to the WLM.



Define a distributed DVIPA and the definitions in the distribution stack using following sub statements for VIPADYNAMIC statement:

- ▶ **VIPADefINE**

Define a Distributed DVIPA.

- ▶ **VIPADISTRIBUTE**

Enables (using DEFINE) the Sysplex Distributor function for a dynamic VIPA (defined on the same stack by VIPADefINE or VIPABACKUP) for which new connection requests can be distributed to other stacks in the sysplex. Specify service port number (for DB2 for z/OS the DRDA port specified in BSDS) and the dynamic XCF address of the target stack to which the distributing DVIPA will forward requests.

Define a back up stack for the distributed DVIPA, in one or more sysplex members in case of failure. Define in the sub statement of VIPADYNAMIC statement on a stack other than your distributed stack:

- ▶ **VIPABACKUP**

Define a back up DVIPA

The representation of the Sysplex Distributor spread across sysplex is intended to show that the Sysplex Distributor is accessible from any system of the sysplex group, but it is a DVIPA which represents a group of members. Sample definitions for our test environment are covered in Chapter 3, "Installation and configuration" on page 69.

**Note:** You may need a dynamic routing feature, such as OMPRoute or RIP enabled, to give full capability of network failover. We included all the IP addresses available in our test system in same subnet to avoid configuring dynamic routing. Be sure you configure your Sysplex Distributor and network according to your environment needs.

## The automatic client reroute operation

Transaction pooling and automatic client reroute (ACR), as well as seamless automatic client reroute (seamlessACR), are enabled by default when sysplex WLB is enabled. This combination allows you to take advantage of the function to reroute a live connection to a different member of a data sharing group.

**Restriction:** Document based on V9.5 FP3 or later.

Connection rerouting is possible because transaction pooling disassociates the connection from the Transport within the DRDA AR client. If one member of DB2 data sharing group has failed, DRDA AR clients knows, and simply dispatches the next transaction request from each connection a transport that is connected to an available member of the data sharing group.

The Type 4 driver and the non-Java-based Data Server Driver give further capability called seamlessACR, where the driver retries the transaction on the new member without notifying the application when possible. When the first SQL in transaction was executing and DB2 server fails, seamlessACR will work and retries the SQL on other available DB2 data sharing members. When sysplex WLB is enabled, seamlessACR is always used using the Type 4 driver. The same applies to the non-Java-based Data Server Driver, where sysplex WLB is enabled seamlessACR is enabled by default. You can disable seamlessACR when the parameter for seamlessACR was explicitly set to turn it off.

**Note:** A seamless failover, as described in the Java application development for IBM data servers documentation at the following Web page is equivalent to seamlessACR described here:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.apdv.java.doc/doc/c0024189.html>

For the DB2 Connect server, ACR is performed when the DB2 Connect server is at the same level as the client or higher, the client can perform seamless failover. Otherwise, the client does not perform seamless failover, and the error SQL30108N is returned to the applications.

The resilience of the DRDA connection is illustrated in Figure 6-4, which shows member D9C1 failing, and the connected workload switching automatically to member D9C2. A detailed examination of the possible application failover scenarios is covered in 6.3, “DB2 failover scenario with and without Sysplex Distributor” on page 259.

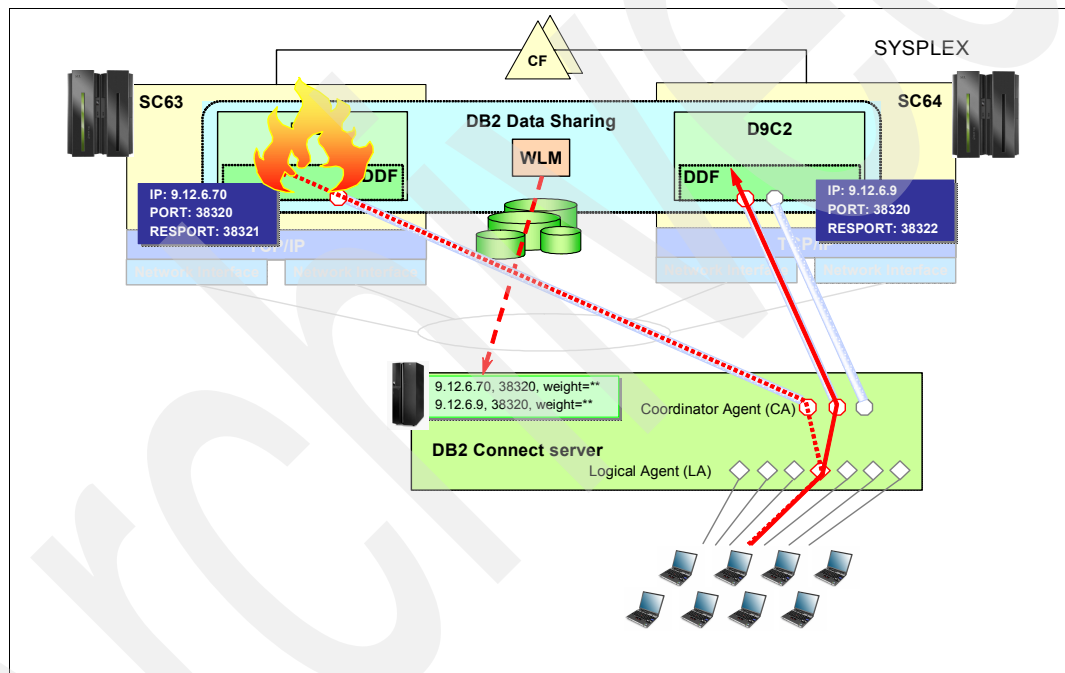


Figure 6-4 Client reroute

### 6.1.6 Scenario with Q-Replication for high availability

Some customers may choose to have multiple DB2 data sharing groups configured for high availability and disaster recovery purposes where the data is copied across using a replication solution as shown in Figure 6-5 on page 247. The IBM Data Server drivers can only be used for workload balancing among members of a DB2 data sharing group. It is not possible to route the workload across DB2 data sharing groups seamlessly just by using the drivers. If an asynchronous replication solution, such as Q-Replication, is used to copy data between two sysplexes, the status of data being replicated is not known or guaranteed and it is recommended to do a user directed switch between the DB2 data sharing groups.

If you have configured multiple copies of application servers where each server directs the workload to another DB2 data sharing group, you can use the external workload router to balance the workload across multi-DB2 data sharing system.

For example, when the application servers is configured to use the Sysplex on the left of Figure 6-5, the workload will be balanced within the data sharing group as shown by the long dashed (red) arrows. The same is true for the application servers configured to use the Sysplex to the right drawn as blue bold arrows. In case of a whole Sysplex failure, users needs to change their setting manually to direct the workload to the other group, as shown by the dotted blue arrows.

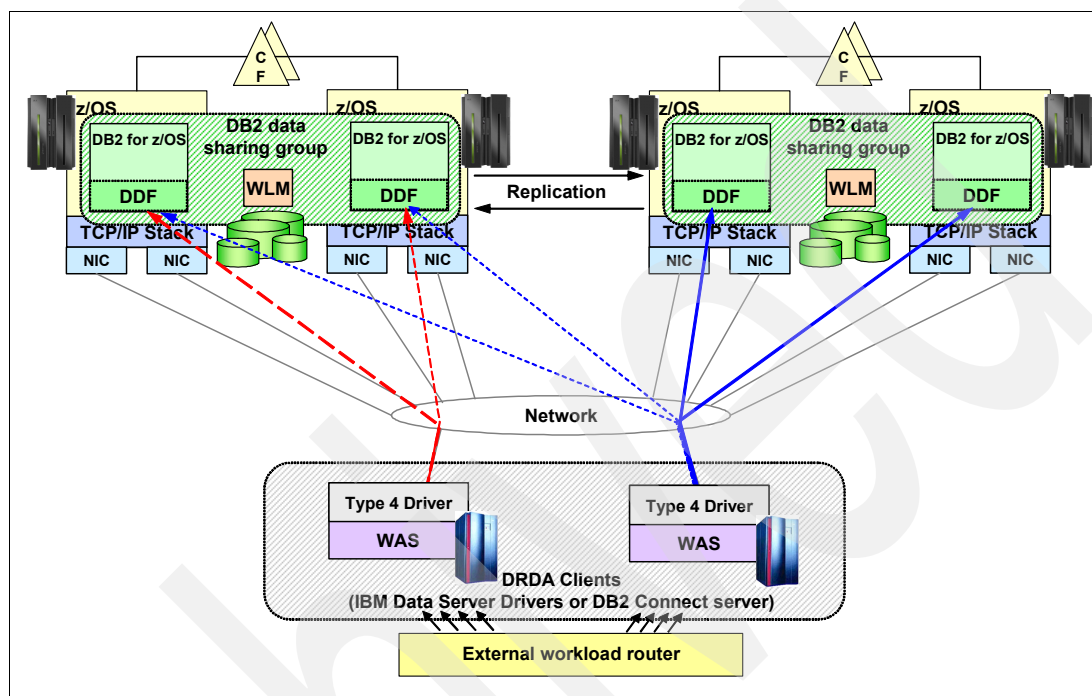


Figure 6-5 Multi-sysplex configuration scenario

## 6.2 Recommendations for common deployment scenarios

The considerations for a resilient infrastructure for distributed access to DB2 for z/OS fall into two categories:

- DB2 for z/OS server resilience

The most resilient DB2 for z/OS servers exploit DB2 data sharing, parallel sysplex, Sysplex Distributor, and DVIPA. The tests in this chapter have shown that such a configuration provides resilience for all scenarios except an in-flight unit of work. Optionally, we will show a DB2 data sharing subsetting scenario.

- The clients and drivers resilience

Depending on the nature of the distributed applications, we consider the scenarios which utilize connection pooling by application servers.

There are several differences in terms of configuring for a resilient infrastructure using DRDA to access DB2 for z/OS. This section shows different ways of configuring each DRDA AR client.

## 6.2.1 DB2 data sharing subsetting

Suppose you want to configure a three member sysplex but you want only two out of those three DB2 data sharing members used for DRDA access. Using a location alias, you can create a subset of DB2 data sharing members. Load distribution for both transactions and connections are performed across the subset. If all members of the subset are down, connection failures will occur even if other members not in the subset are started and capable of processing work. By designating subsets of members, you can perform the following tasks:

- ▶ Limit the members to which DRDA AR clients can connect. System and database administrators might find this useful for any number of purposes.
- ▶ Ensure that initial connections are established only with members that belong to the specified subset. Without subsets, requesters can make initial and subsequent connections to any member of the data sharing group.
- ▶ Provide requesters with information about only those members in the subset.

Figure 6-6 shows test configuration used in our test environment. Configuring DB2 data sharing subsetting is straightforward. Use the Change Log Inventory Utility(DSNJU003) to set DDF ALIAS to subset member of BSDS.

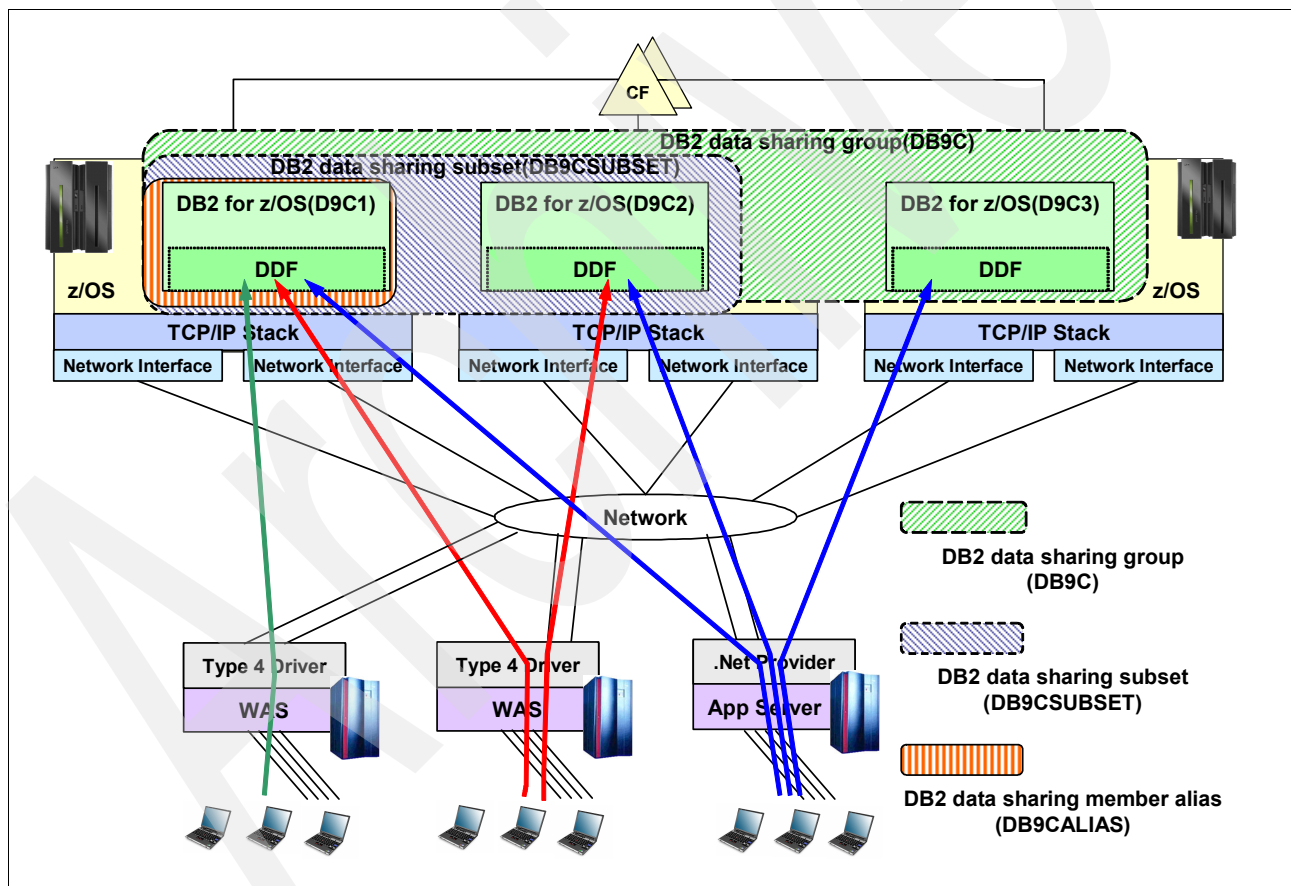


Figure 6-6 DB2 data sharing subsetting

Our configuration has location name DB9C (which consists of all three DB2 data sharing members), location alias DB9CSUBSET (which consists of D9C1 and D9C2), and location alias DB9CALIAS (which consists of D9C1). Each location alias listens to different port numbers.

Assume DRDA AR clients all configured for sysplex WLB, each server connected to a different location name and port number. As Figure 6-6 on page 248 shows, each DRDA AR client only distribute the workload to applicable members.

**Important:** We do not recommend defining a DB2 data sharing subset that consists of only one member, since that would be the same as connecting to a specific IP address and port number without gaining the availability benefits of DB2 data sharing. To have a resilient infrastructure, we recommend defining a data sharing subset with at least two members.

The only reason to have a single member subset is if your requirement to restrict workload from one or more members exceeds your requirements for availability.

APAR PK80474, also mentioned at 6.1.2, “z/OS WLM in DRDA workload balancing” on page 234, has enhanced DB2 for z/OS V8 to list all location alias names and port numbers, which was made first available in DB2 9 for z/OS.

You can use this function to define one subset of a data sharing group to support distributed access and another subset to support batch workloads, for example.

## 6.2.2 Application Servers

This section gives practical deployment scenario with recommendation for Java-based application server using Type 4 driver and non-Java-based application server using the non-Java-based IBM Data Server Driver or Client. Sysplex Distributor should always be used in conjunction with sysplex WLB.

### DB2 for z/OS server resilience

Regarding the DB2 for /OS server, use a DB2 data sharing group with Sysplex Distributor and DVIPA, to provide maximum resilience of the DB2 for z/OS server.

### Network resilience

Regarding the TCP/IP network outside the sysplex, alternate routing should be configured in the TCP/IP network to provide alternate routes to the DB2 server.

### WebSphere Application Server connectivity

When using Java-based application servers, such as WebSphere Application Server to connect to DB2 data sharing group, you can take advantage of the DB2 data sharing group using Type 4 driver without going through DB2 Connect Server.

The minimum level of requirement for Type 4 driver is 2.7.xx or later. Example 6-5 shows command output from system with DB2 Connect V9.5 FP3. If not already defined to system, the classpath to Type 4 driver module needs be defined or given to a Java Virtual Machine (JVM) through Java option when issuing commands.

#### *Example 6-5 Verify level of Type 4 driver*

```
$java com.ibm.db2.jcc.DB2Jcc -version
IBM DB2 JDBC Universal Driver Architecture 3.53.70
```

**Note:** Our example showed IBM DB2 JDBC Universal Driver, which was the older name for Type 4 driver, you will see new name in a message, when nnn is 4.0 or later.

```
C:\DDF\test\javatests>java com.ibm.db2.jcc.DB2Jcc -version
IBM Data Server Driver for JDBC and SQLJ 4.8.23
```

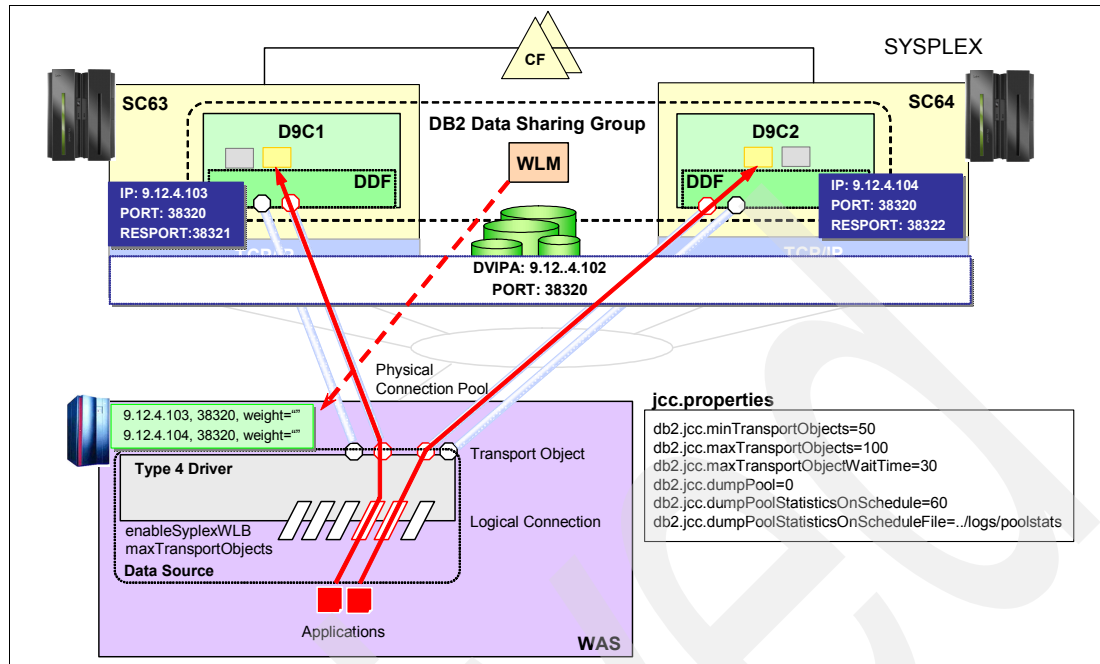


Figure 6-7 Java application server scenario configuration using WebSphere Application Server

To set up your sysplex WLB with Type 4 driver, you have two places to configure your options. One is datasource properties where you enable sysplex WLB and limit your number of transports, and another is Type 4 driver configuration properties where you set global limit of transports and set your monitoring.

Following are the lists of properties to enable and limits:

- ▶ IBM Data Server Driver for JDBC and SQLJ data source properties
  - enableSysplexWLB
  - enableConnectionConcentrator
  - maxTransportObjects
- ▶ IBM Data Server Driver for JDBC and SQLJ configuration properties
  - db2.jcc.minTransportObjects
  - db2.jcc.maxTransportObjects
  - db2.jcc.maxTransportObjectIdleTime
  - db2.jcc.maxTransportObjectWaitTime
  - db2.jcc.dumpPool
  - db2.jcc.dumpPoolStatisticsOnSchedule
  - db2.jcc.dumpPoolStatisticsOnScheduleFile

Table 6-3 on page 251 shows recommended settings for common java application environments for high availability. Start from recommendation and tune each properties as you monitor.

Table 6-3 Recommendation for common deployment

Setting	Description	Default	Recommended
enableSysplexWLB	Enables Sysplex WLB	false	true (to enable WLB)
enableConnectionConcentrator	Enables Connection Concentrator	Defaults to true when enableWLB is true	true
maxTransportObjects	Specifies the maximum number of transports per DataSource object.	-1 (limit to db2.jcc.maxTransport Objects)	50–100 range
db2.jcc.minTransportObjects	Specifies the minimum number of transports in the transport pool	0 (no global transport pool)	25–50 range (about half of maxTransportObject)
db2.jcc.maxTransportObjects	Specifies the max number of transports in the transport pool	-1 (implies unlimited)	1000
db2.jcc.maxTransportObjectIdleTime	Specifies the time in seconds that an unused transport object stays in a global transport object pool before it can be deleted from the pool. Transport objects are used for workload balancing.	60	>0
db2.jcc.maxTransportObjectWaitTime	Specifies the maximum amount of time in seconds that an application waits for a transport object if the db2.jcc.maxTransportObjects value has been reached.	-1 (implies unlimited)	30
enableSeamlessFailover	Enables Seamless automatic client reroute with failover	When DB2 for z/OS is the server and enableSysplexWLB is true, enableSeamlessFailover is automatically enabled.	DB2BaseDataSource.YES (optional)

**Note:** Recommended values may vary depending on your installation and how your applications are implemented.

### Setting the Type 4 driver configuration properties

Prepare the configuration file using your preferred editor. Example 6-6 shows the contents of the configuration file.

*Example 6-6 Example of configuration properties file*

```
db2.jcc.minTransportObjects=50
db2.jcc.maxTransportObjects=1000
db2.jcc.maxTransportObjectWaitTime=30
db2.jcc.dumpPool=0
db2.jcc.dumpPoolStatisticsOnSchedule=60
db2.jcc.dumpPoolStatisticsOnScheduleFile=/home/db2/logs/poolstats
```

**Note:** If you are configuring the global transport object pool monitor, be sure you have a sufficient authority to access a file and directory. With the setting in the example, you will get logs filled every one minutes. Make sure you have sufficient space and clean up frequently.

Set your configuration file through JVM properties. For WebSphere Application Server environments, add new properties for JVM by navigating to **Application servers** → **your copy of application server** → **Process Definition** → **Java Virtual Machines** → **Custom Properties**. Figure 6-8 shows an example setting configuration file named “jcc.properties”.

Application servers > server1 > Process Definition > Java Virtual Machine > Custom Properties > New

Specifies an arbitrary name-value pair. The value is a string that can set internal system configuration properties.

Configuration

**General Properties**

\* Name: db2.jcc.propertiesFile

\* Value: /home/db2isnt1/jcc.propertie

Description: Properties file for jcc

Apply OK Reset Cancel

Figure 6-8 Setting Type 4 driver configuration properties file to WebSphere Application Server



**Note:** APAR PK41236 added sysplex WLB support for Type 4 driver with KEEP\_DYNAMIC(YES) specification to DB2 V8 and 9. Until this change, KEEP\_DYNAMIC(YES) prevents thread from becoming inactive, which prevents DRDA AR clients from re-using connections.

To implement this change, the server returns information about transaction boundaries to tell the Type 4 driver if disconnect happens after COMMIT/ROLLBACK, the client will not be able to connect to the original server. If there are no held cursors, declared global temporary tables, etc., and the only thing preventing clients from reusing connections was KEEP\_DYNAMIC(YES), no errors are returned and the SQL statement flows to a different DB2 data sharing member, even if some statement needs to be re-prepared to execute. In the case where the client does not get a disconnect failure, behavior does not change, so users can have the benefit of having KEEP\_DYNAMIC(YES).

The Type 4 driver (version 3.51.x/4.1.x or later) has a feature which supports the specification of the datasource properties "keepDynamic" and "enableSysplexWLB" to be enabled for connections to a DB2 for z/OS data sharing group. When both properties are enabled, the changes described will take effect.

## .NET Provider/CLI Driver

For non-Java-based application servers, there are two different possibilities. For .NET or CLI environment, you can choose from either one of the two following packages with DB2 Connect license:

- ▶ DB2 Connect client/server
- ▶ The non-Java-based IBM Data Server Driver or Client

**Note:** If you installed the non-Java-based Data Server Driver, you will not have a database directory. So configure dsn alias in your data server driver configuration file, ds2dsdriver.cfg. For any of the clients with a database directory, such as DB2 Connect server/client, the db2dsdriver.cfg file will not be used for alias lookup. Optionally, you can also give server information to your application through connection attributes, similar to Type 4 driver configuration.

Figure 6-9 on page 254 shows a sample configuration for .NET Provider environment. The logical configuration is similar to the Java environment except for the data server driver configuration file, where sysplex WLB and other configuration options such as number of transports are set.

The sysplex WLB happen within a process where logical connection of application and transport are separated. The application server configuration for .NET Provider is similar to WebSphere Application Server environments where you can take full advantage of the sysplex WLB.

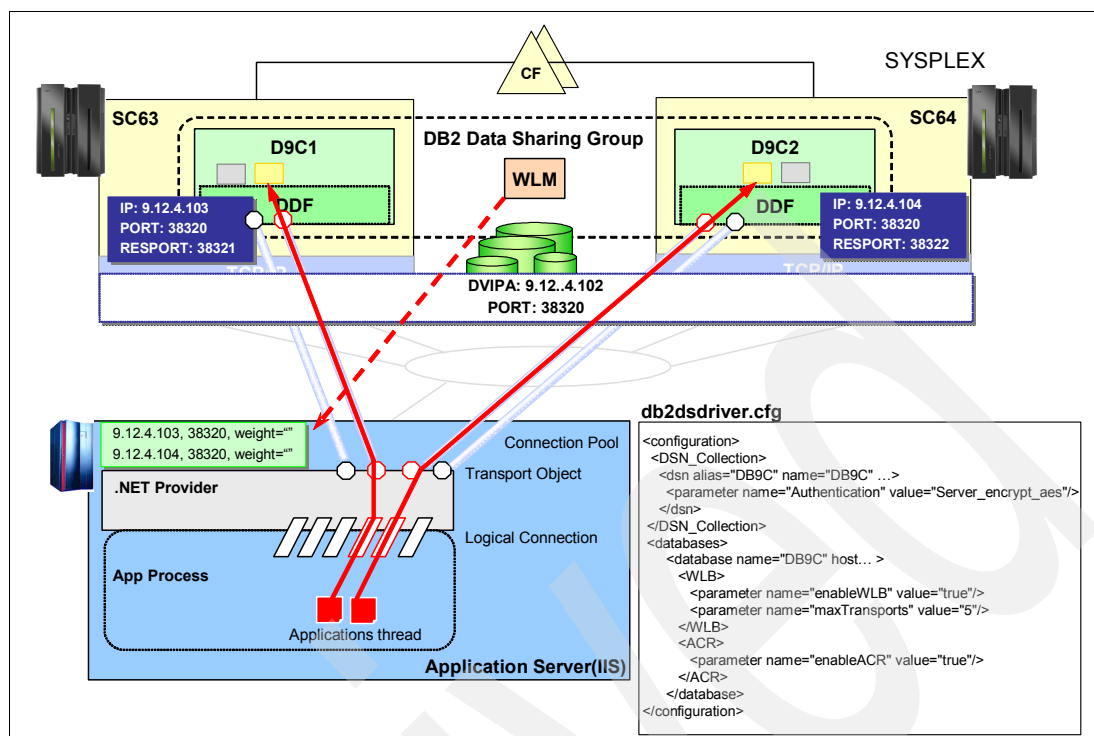


Figure 6-9 Non-Java-based application server scenario configuration using .NET

The transaction-pooling and seamlessACR should always be enabled when using sysplex WLB. Table 6-4 lists the recommended settings for the highest availability. Those parameters should be pass to the non-Java-based Data Server Driver through configuration file, db2dsdriver.cfg.

Table 6-4 Recommended settings for the non-Java-based IBM Data Server Driver

Setting	Description	Default	Recommended
enableWLB	Enables sysplex WLB	false	Set to true to enable WLB
maxTransports	Specifies the max number of transports in the transport pool	-1 (implies unlimited)	Start from 50-100 range
maxTransportIdleTime	Specifies the maximum elapsed time in number of seconds before an idle transport is dropped	600	600
maxTransportWaitTime	Specifies the number of seconds that the client will wait for a transport to become available	-1(unlimited)	30
maxRefreshInterval	Specifies the maximum elapsed time in number of seconds before the server list is refreshed	30	30
enableACR	Enables automatic client reroute	Defaults to true when enableWLB is true	true (optional)

Setting	Description	Default	Recommended
enableSeamlessACR	Enables Seamless automatic client reroute with failover	When DB2 for z/OS is the server and enableACR is true, enableSeamlessACR is automatically enabled.	true (optional)

Example 6-7 shows a configuration file for the non-Java-based IBM Data Server Driver.

*Example 6-7 Sample setting for db2dsdriver.cfg*

```

<configuration>
  <DSN_Collection>
    <dsn alias="DB9C" name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="Authentication" value="Server_encrypt_aes"/>
    </dsn>
  </DSN_Collection>
  <databases>
    <database name="DB9C" host="wtsc63.itso.ibm.com" port="38320">
      <parameter name="CurrentSchema" value="PAOLOR7"/>
      <parameter name="DisableAutoCommit" value="1"/>
      <parameter name="ProgramName" value="PID"/>
      <WLB>
        <parameter name="enableWLB" value="true"/>
        <parameter name="maxTransports" value="100"/>
        <parameter name="maxTransportWaitTime" value="30"/>
        <parameter name="maxTransportIdleTime" value="600"/>
        <parameter name="maxRefreshInterval" value="30"/>
      </WLB>
      <ACR>
        <parameter name="enableACR" value="true"/>
      </ACR>
    </database>
  </databases>
  <parameters>
    <parameter name="CommProtocol" value="TCPIP"/>
  </parameters>
</configuration>

```

The following applies to clients with a database directory, such as DB2 Connect server/client connecting to DB2 for z/OS:

- ▶ The db2dsdriver.cfg file will not be used for “alias lookup”. You should catalog your database using the catalog command before making a connection or should provide the connection attributes in the applications.
- ▶ To enable sysplex WLB, you need to have ‘,,,,,SYSPLEX’ parameter specified in the Database Connection Services directory.

**Tip:** If you are migrating from DB2 Connect server/client to the non-Java-based IBM Data Server Driver or Client, you can use the following command to read the current database directory information to create the basic configuration file. You will need to customize the configuration file to enable the workload balancing functions.

```
$ db2dsdcfgfill
SQL01535I The db2dsdcfgfill utility successfully created the db2dsdriver.cfg
configuration file.
```

',,,,,SYSPLEX' keyword is equivalent to the enableWLB parameter that should be set through the db2dsdriver.cfg file. This parameter will not be set automatically through the db2dsdcfgfill command. To enable Sysplex Workload balancing, you need to set both the enableWLB and enableACR parameters, which is equivalent to enabling the connection concentrator function in DB2 Connect server.

For Windows platform users, the configuration file will be created in cfg directory under instance home.

When your DB2 profile registry look as follows:

```
C:\>db2set -all
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[i] DB2INSTPROF=C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY3
[i] DB2COMM=TCPIP
[g] DB2_EXTSECURITY=NO
[g] DB2SYSTEM=LENOVO-B6AFDE0A
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS03
```

The configuration file will be created at following location:

```
C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY3\DB2\cfg
```

If you do not have the cfg directory under your instance home, the db2dsdcfgfill command will fail. Create the directory before executing the command.

### 6.2.3 Distributed three-tier DRDA clients

This section gives practical recommendations for a distributed three-tier DRDA configuration.

#### DB2 for z/OS server resilience

Regarding the DB2 for z/OS server, use a DB2 data sharing group with Sysplex Distributor and DVIPA, to provide maximum resilience of the DB2 for z/OS server.

Additionally, implement connection concentrator (transaction pooling) in the DB2 Connect.

#### Network resilience

Regarding the TCP/IP network outside the sysplex, alternate routing should be configured in the TCP/IP network to provide alternate routes to the DB2 server.

## DB2 Connect server (gateway configuration)

When you are configuring to use application servers, there should be no need to add a DB2 Connect server in your installation, since that will just introduce an additional single point of failure in your system.

When all the client applications (such as MS® Excel®, and so forth) come in from each client user's notebook computer, DB2 Connect server is probably still an option to do the traffic control to limit the number of actual connections to DB2 for z/OS in conjunction with Connection Concentrator for transaction pooling.

Figure 6-10 shows how DB2 Connect server acts as a gateway when applications connect to DB2 for z/OS data sharing group.

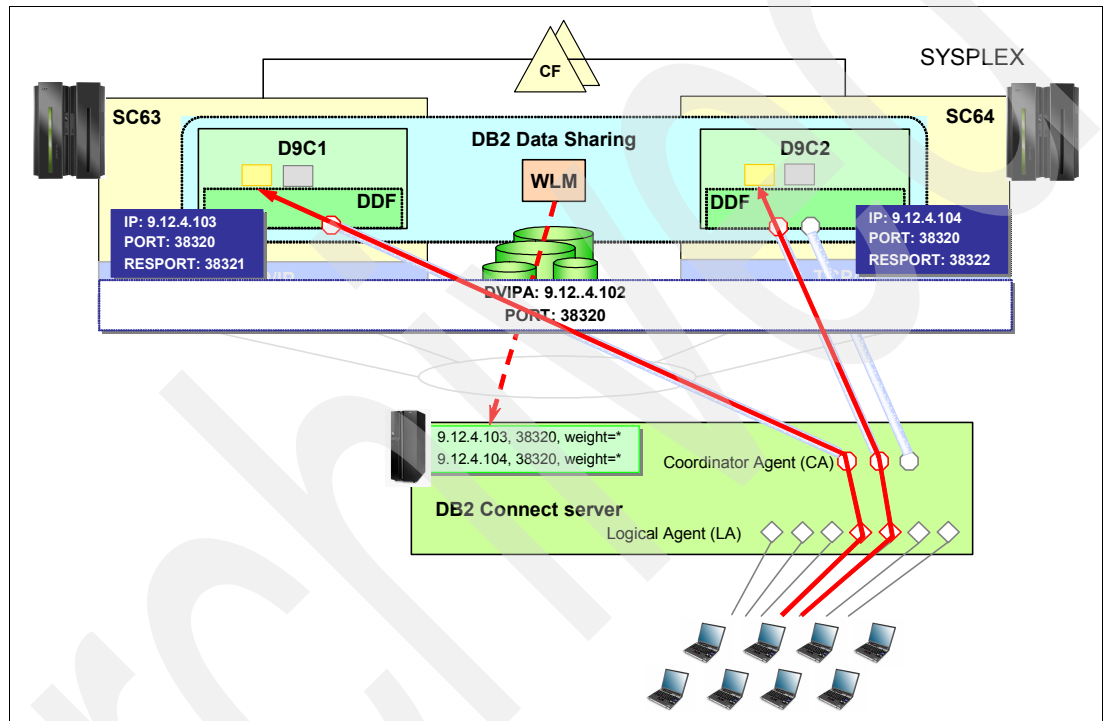


Figure 6-10 Distributed three tier DRDA clients scenario

When using DB2 Connect server, you need to catalog the database, node, and DCS directory. For example, catalog the database using Distributed DVIPA(9.12.4.102) to connect to DB2 data sharing group. The initial connection will be made as long as one of DB2 data sharing member exists.

Cataloging a connection to the DB2 data sharing group is as follows:

```
catalog tcpip node ZOSNODE remote 9.12.4.102 server 38320
catalog database DB9C as DB9C at node ZOSNODE authentication server_encrypt
```

Catalog DCS directory is optional, since Sysplex support is set by default. If you cataloged your database directory using location name, you do not need to catalog DCS directory. If you used a location name longer than 8 bytes, you need to catalog DCS directory with the ',,,,,SYSPLEX' option.

```
catalog DCS database DB9C as DB9C parms ',,,,,SYSPLEX'
```

Table 6-5 on page 258 lists recommended settings for common environments. You can start from the recommendation and tune your settings as you monitor your environment.

Table 6-5 Recommended settings for DB2 Connect server configuration

Setting	Description	Default	Recommended
parms ',,,,,SYSPLEX'	Enables Sysplex WLB	true (if DCS directory node not defined)	Set parms in sixth field to enable sysplex WLB
max_coordagents	Specifies the max number of agents (transports) in the agent pool	AUTOMATIC	100
max_connections	Specifies the max number of connections allowed to connect to DB2 Connect server	AUTOMATIC	101 (max_coordagents +1) to turn on connection concentrator
num_poolagents	Specify the maximum size of the idle agent pool	100, AUTOMATIC	50
num_initagents	Specify the initial number of idle agents that are created in the agent pool at DB2START time	0	50
DB2_MAX_CLIENT_CONNRETRIES	The maximum number of connection retries attempted by automatic client reroute.	30(DB2_CONNRETRIES_INTERVAL not set), 10(DB2_CONNRETRIES_INTERVAL set)	not set
DB2_CONNRETRIES_INTERVAL	The sleep time between consecutive connection retries, in number of seconds.	30	not set
DB2TCP_CLIENT_CONTIMEOUT	Specifies the number of seconds a client waits for the completion on a TCP/IP connect operation	When DB2 for z/OS is the server and enableACR is true, enableSeamlessACR is automatically enabled.	10

**Note:** Settings based on DB2 Connect V9.5 FP3

A sample settings for connection concentrator and ACR using the recommendation shown in Table 6-5 is as follows:

```
$ db2 update dbm cfg using MAX_CONNECTIONS 101 MAX_COORDAGENTS 100 MAXAGENTS 100
NUM_POOLAGENTS 50 NUM_INITAGENTS 50
$ db2set DB2TCP_CLIENT_CONTIMEOUT=10
B
```

**Tip:** If you are running an application on the same host as the DB2 Connect server, and if you want the application to go through DB2 Connect, you need to set the DB2 registry variable:

```
db2set DB2CONNECT_IN_APP_PROCESS=NO
```

### DB2 Connect server failover (DRDA three-tier client resilience)

Whenever DB2 Connect server crashes, all clients connected through that DB2 Connect server to DB2 for z/OS receive a communications error, which terminates the connection resulting in an application error. In cases where availability is important, you should have implemented either a resilient set up or the ability to fail the server over to a standby or

backup node. In either case, the DB2 client code attempts to re-establish the connection to the original server, which might be running on a failover node, or to a new server.

You can set your alternate server to DB2 Connect server using the UPDATE ALTERNATE SERVER FOR DATABASE command as follows:

```
update alternate server for database altserver using hostname host2 port 54320
```

**Restriction:** If you configured the two phase commit processing using SPM log of DB2 Connect server, failover nodes need to take over the failed SPM log to recover all the transactions. It is recommended not to use SPM log (two phase commit processing) when configured with DB2 Connect server client reroute.

The following environment variables can be used to influence how client reroute behaves for client applications.

- ▶ DB2TCP\_CLIENT\_RCVTIMEOUT
- ▶ DB2\_MAX\_CLIENT\_CONNRETRIES

Refer to *DB2 9.1 Administration Guide: Implementation*, SC10-4221 for a description of these environment variables.

## 6.3 DB2 failover scenario with and without Sysplex Distributor

The role of z/OS Sysplex Distributor is to increase the resilience and recovery of applications in System z parallel sysplex. DB2 for z/OS takes advantage of the Sysplex Distributor, as discussed earlier in this chapter.

In this section we show some practical scenarios of the failover resilience of DRDA connectivity to a DB2 for z/OS data sharing group, with or without Sysplex Distributor. The intent is to demonstrate the effectiveness of Sysplex Distributor.

### 6.3.1 Configuration for scenarios

The configuration used for testing failover scenarios with Sysplex Distributor is represented in Figure 6-11 on page 260. Test was done with and without Sysplex Distributor. The case without Sysplex Distributor was performed specifying member-specific VIPA address at the DRDA AR client.

- ▶ DB2 for z/OS data sharing group is configured with/without Sysplex Distributor.
  - Sysplex Distributor: Configured with Distributed DVIPA with 9.12.4.102
  - Two members: D9C1 and D9C2 at 9.12.4.103 and 9.12.4.104, respectively
  - Location name: DB9C
- ▶ Four DRDA Application Requestors are configured.
  - DB2CCL: DB2 Connect Personal Edition V9.5 FP3 for Windows (no connection concentrator)
  - DB2CSV: DB2 Connect Enterprise Edition V9.5 FP3 for AIX (direct connection with CLI driver)
  - JCC: The Type 4 driver, sysplex WLB
  - DB2zOS driver: DB2 9 for z/OS

**Note:** Test scenarios are all configured with connection concentrator, except for the DB2CCL scenario. Product recommendation is to configure connection concentrator if you are using sysplex awareness. The Type 4 driver and CLI Driver will set connection concentrator as default if you configure sysplex awareness.

You get Sysplex WLB capability by default without the need to catalog your DCS directory. You need to configure sysplex WLB in your DCS directory if you catalog the DCS directory.

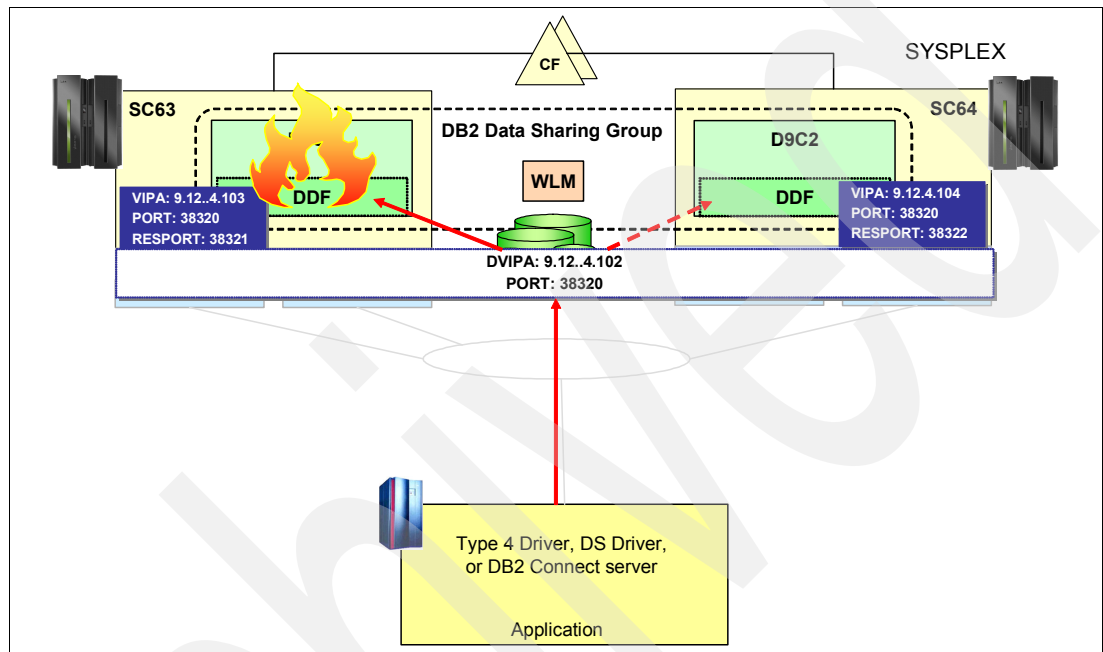


Figure 6-11 Failover test scenario with Sysplex Distributor

The failure that is tested is an outage of D9C1, which is one member of the data sharing group. The outage is achieved by cancelling the IRLM with the SDSF command or stopping DDF MODE(FORCE) at DRDA AS site:

```
/F D9C1IRLM,ABEND,NODUMP
```

When D9C1 is cancelled or DDF stopped, any connection to D9C2 continues unaffected. The purpose of these tests is to confirm what happens to an application that is connected (or trying to connect) to D9C1.

### 6.3.2 Application states for scenarios

In order to test failover resilience, it is necessary to build application test cases that represent all the possible states that an application can be in when a failure occurs. There are five possible application connection states that can be in effect when D9C1 failed. The application states (AS1 to AS5) used are:

#### ► AS1

The application is not connected to DB2 when member D9C1 fails. The requester has not established a connection to D9C1, and hence does not have member information. The application then issues CONNECT TO DB9C



► AS2

The application is not connected to DB2 when D9C1 fails. A DRDA client has previously established a connection to D9C1, which has returned the member information to a client. The client then issues CONNECT TO DB9C.

► AS3

The DRDA client is connected to D9C1, but has no unit of work in progress, when D9C1 fails. The application then issues another SQL statement, PREPARE (followed by DECLARE, OPEN, FETCH, CLOSE, COMMIT).

► AS4

The DRDA client is connected to D9C1, and has just issued a COMMIT, but has a CURSOR WITH HOLD open when D9C1 fails. The application then issues another SQL call against the held cursor:

- FETCH
- CLOSE
- COMMIT

► AS5

When the client is connected to D9C1 and has an in-flight unit of work (insert, plus open cursor with one row fetched) when D9C1 fails. The application then issues another SQL statement:

- INSERT
- COMMIT

The application that runs as a DB2 z/OS AR is an open with hold cursor with one row fetched, when D9C1 fails. The application then issues other SQL statements:

- FETCH
- CLOSE
- COMMIT
- PREPARE
- ...

### 6.3.3 Results without Sysplex Distributor

Table 6-6 shows a matrix of the test results without Sysplex Distributor. The table lists the SQLCODES for the SQL statements attempted after member D9C1 failed.

*Table 6-6 Test results after D9C1 failed without Sysplex Distributor*

Application state	DB2CCL	DB2CSV	JCC	DB2zOS
AS1	SQLCODE: -30081	SQLCODE: -30081	SQLCODE: -4499	SQLCODE: -30081
AS2	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0
AS3	SQLCODE: -30081	SQLCODE: 0	SQLCODE: 0	SQLCODE: -30081
AS4	Fetch: 0 Close: -1024	Fetch: 0 Close: -30108	Fetch: 0 Close: -30108	Fetch: 0 Close: -918
AS5	Insert: -30081 Commit: -1024	Insert: -30108 Commit: 0	Insert: -30108 Commit: 0	Fetch: 0 Close: -918

**Tip:** SQLCODE: -30108 means that the query fails due to the member to which you are connected not being available. Sysplex awareness does a client reroute. If you capture -30108, the executing UOW work has rollback, but you can rerun your UOW without reconnecting unless locks from a previous execution remain as retained locks.

In our test scenario with seamless failover no error is returned to the application.

### Application state 1

The application is not connected when member D9C1 fails. The DRDA AR client has not established a connection to D9C1, and hence does not have member information. The application then issues CONNECT TO D9C1.

- ▶ For all DRDA client scenarios
  - Without Sysplex Distributor, a DB2 Connect configuration must have been started, and must have made at least one successful connection to the DB2 data sharing group, to have the member list information that allows connections to be routed to alternate members of the data sharing group.
  - Application state 1 defines the case where DRDA client had not been previously started before the D9C1 failure, and hence any SQL Connect from any (cold-started) DRDA client fails with the generic connection failure SQLCODE -30081.

- ▶ For DB2 for z/OS

DB2 for z/OS, acting as a DRDA AR, also makes use of the member list information. As with DRDA client in application state 1, and without Sysplex Distributor, DB9A (DRDA AR) has not retrieved the member list information, and any SQL CONNECT fails with SQLCODE -30081.

### Application state 2

The application is not connected to DB2 when D9C1 fails. DB2 Connect has previously established a connection to D9C1, which has returned the member information to DRDA AR client. The application then issues CONNECT TO D9C1.

- ▶ For all DRDA client scenarios

Application state 2 defines the case where DRDA AR client had been previously started, and connected to D9C1 before the D9C1 failure. In this case, DRDA AR client will have retrieved the member list information, and can reroute the connection request to D9C2 with SQLCODE 0.

- ▶ For DB2 for z/OS

Application state 2 defines the case where DB9A client had been previously started and connected to D9C1 before the D9C1 failure. In this case, DB9A will have retrieved the member list information, and reroutes the connection request to D9C2 with SQLCODE 0.

### Application state 3

The application is connected to D9C1, but has no unit of work in progress when D9C1 fails. The application then issues PREPARE (followed by DECLARE, OPEN, FETCH, CLOSE, COMMIT).

- ▶ For DB2 Connect client (no connection concentrator)

Once connected to member D9C1, even if no unit of work is in-flight, the connection is exclusively tied to member D9C1. If member D9C1 fails, then the connection is lost. The application does not get any notification of the lost connection until the next time it tries to issue an SQL statement. In this test, it attempts an SQL SELECT, and receives an SQLCODE -30081.

- ▶ For all other DRDA AR client scenarios

With transaction pooling, the connection is transferred automatically to D9C2 because the connection is disassociated from transports, and is not exclusively tied to any individual member. The new unit of work is started on a DBAT in D9C2, totally transparent to the program.

- ▶ For DB2 for z/OS

DB2 for z/OS as a DRDA AR does not have the transaction pooling functionality that allows a second unit of work to be rerouted to an alternate DB2 for z/OS sysplex member. DB2 returns an SQLCODE -30081.

### Application state 4

The application is connected to DB2, and has just issued a COMMIT, but has a CURSOR WITH HOLD open when D9C1 fails. The application then issues the following commands:

- ▶ FETCH
- ▶ CLOSE
- ▶ COMMIT
- ▶ PREPARE (followed by DECLARE, OPEN, FETCH, CLOSE, COMMIT)

- ▶ For DB2 Connect client (no connection concentrator)

- The FETCH succeeds because the blocked cursor is cached in the DB2 Connect block cache (as set by RQRIOBLK).
- The CLOSE and COMMIT both fail with SQLCODE -1224. This SQLCODE is not as helpful as it could be. It is chosen by DB2 Connect because the connection agent is already established and working with the cursor, but can no longer access the cursor.
- The PREPARE is not related to the existing cursor, and DB2 Connect returns a more helpful -1024 (a database connection does not exist).

- ▶ For all other DRDA AR client scenarios

- The FETCH succeeds because the blocked cursor is cached in the DB2 Connect block cache (as set by RQRIOBLK).
- The CLOSE fails with SQLCODE -904, resource unavailable.
- The COMMIT and PREPARE succeed because the automatic transfer to D9C2 has completed successfully, and further SQL statements can now be executed.

- ▶ For DB2 for z/OS

- The FETCH succeeds because the blocked cursor is cached by DB2 for z/OS DRDA blocking.
- The PREPARE fails with SQLCODE -918. The SQL statement cannot be executed because a connection has been lost.

## Application state 5

The application is connected to DB2, and has an in-flight unit of work (select and update scenarios are both tested), when D9C1 fails. The application then issues the following commands:

- ▶ INSERT
- ▶ COMMIT
- ▶ For DB2 Connect client (no connection concentrator)
  - The INSERT fails with SQLCODE -30081.
  - The COMMIT returns -1024. A database connection does not exist.
- ▶ For all other DRDA client scenarios
  - The INSERT fails with SQLCODE -30108, connection failed but the client re-established the connection.
  - The COMMIT succeeds because connection was automatically re-route to D9C2 and completed successfully, and further SQL statements can now be executed.
- ▶ For DB2 for z/OS
  - The FETCH succeeds, because the blocked cursor is cached by DB2 for z/OS DRDA blocking.
  - The PREPARE fails with SQLCODE -918. The SQL statement cannot be executed because a connection has been lost.

### 6.3.4 Results with Sysplex Distributor

The execution results with Sysplex Distributor are summarized in a matrix of SQLCODEs, as shown in Table 6-7.

Table 6-7 Test results after D9C1 failed with Sysplex Distributor

Application State	DB2CCL	DB2CSV	JCC	DB2zOS
AS1	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0
AS2	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0	SQLCODE: 0
AS3	SQLCODE: -30081	SQLCODE: 0	SQLCODE: 0	SQLCODE: -30081
AS4	Fetch: 0 Close: -1024	Fetch: 0 Close: -30108	Fetch: 0 Close: -30108	Fetch: 0 Close: -918
AS5	Insert: -30081 Commit: 0	Insert: -30108 Commit: 0	Insert: -30108 Commit: 0	Fetch: 0 Close: -918

The table clearly shows one difference with respect to resilience: If DB2 Connect has not been started, connections that are cataloged against the failed data sharing group member now succeed.

## 6.4 Migration and coexistence

DB2 Connect server, the Type 4 driver, and the CLI driver or Client all provide sysplex WLB, which supports infrastructure fault-tolerance. As described through this chapter, DB2 Connect server spreads the workload amongst like DB2 for z/OS subsystems that are members of the DB2 data sharing group.

However, in DB2 for z/OS V7 and V8 coexistence environment, the Type 4 driver and DB2 Connect server connection concentrator connections are downgraded, as APAR PK05198. For example, when DB2 Connect server makes a connection to DB2 for z/OS V8 member, connections are only balanced across the DB2 for z/OS V8 members and DB2 for z/OS V7 members are not used.

Similar case was reported in migration to New Function Mode in APAR PK06697, where connections cannot be routed to subsystems that are in different mode to the original connection created to the DB2 data sharing group. This support will improve a DB2 data sharing migration.

In conjunction to changes made against DB2 Connect server FixPack 10, DB2 for z/OS V8 is providing a more effective approach. If DB2 Connect server opens a connection to DB2 for z/OS V8 Compatibility Mode, the DB2 for z/OS V8 member now returns connection s information indicating only V7 functionality.

**Note:** This supports requires APAR PK05198 and PK06697 installed in conjunction with DB2 Connect V8 FixPack 10 or later.

DB2 Connect V8.2 is no longer in service as of April 30, 2009. All other descriptions in this book are related to DB2 Connect V9.5 FixPack 3 or later.

Archived



## Part 4

# Performance and problem determination

This part contains the following chapters:

- ▶ Chapter 7, “Performance analysis” on page 269
- ▶ Chapter 8, “Problem determination” on page 323

Archived



## Performance analysis

In a distributed architecture environment, having knowledge of the factors affecting the performance of the application is crucial. By identifying these factors, you can plan to tune them during the system and application design.

Performance tuning is a coordinated work of many functional teams in an organization (such as the database administration group, system programmers, network engineers, and application teams.) Keeping in mind that the main purpose of this chapter is to highlight the factors that affect performance, an attempt has been made to clarify these functional areas so that the teams can work in parallel on performance issues.

Traces are important for performance analysis as they can provide the information that is required for understanding and creating a performance profile of a given application or infrastructure. Traces are described in Chapter 8, “Problem determination” on page 323.

This chapter contains the following sections:

- ▶ “Application flow in distributed environment” on page 270
- ▶ “System topics” on page 271
- ▶ “Checking settings in a distributed environment” on page 294
- ▶ “Obtaining information about the host configuration” on page 307

## 7.1 Application flow in distributed environment

Organizations today are able to serve several customers from different parts of the world. Even a single organization has branches in several cities and countries. The applications built to support their business need to be performing to specification and scalable to a large numbers of users.

In Chapter 2, “Distributed database configurations” on page 33, we described the most commonly used configurations for implementing DB2 for z/OS in a distributed database business environment. In this chapter we consider 2- and 3-tier configurations.

In configurations where applications connect directly to the server, such as clients with Data Server Drivers, a gateway is not required between the DRDA requester and the server. This constitutes a 2-tier configuration. See Figure 7-1.

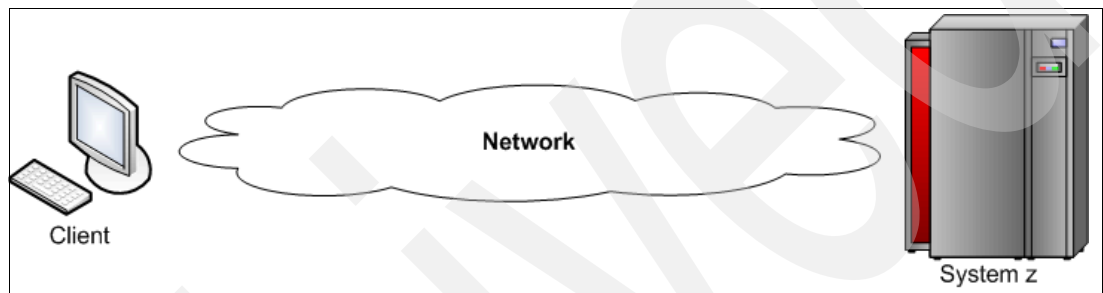


Figure 7-1 2 tier architecture representation

Figure 7-2 illustrates an 3-tier configuration. The clients can be thin or fat. The clients connect to middle-tier servers, namely, Web servers and application servers. The function of the Web servers is to handle the Hypertext Transfer Protocol (HTTP) requests and redirect them to application servers. The application servers handle the business process logic.

In a DRDA environment, to get the data from the host, often communication gateways coordinate with the host for the retrieval and update of information. In this scenario, the middle-tier communication server often used is DB2 Connect Enterprise Edition.

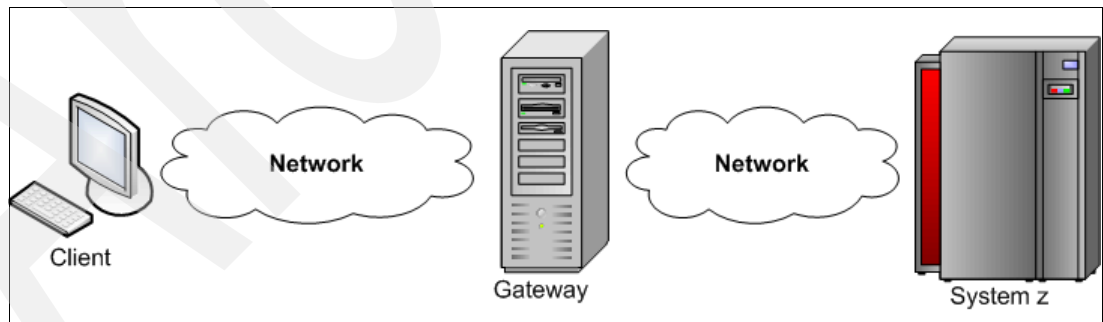


Figure 7-2 3-tier architecture representation

## 7.2 System topics

In this section we discuss the following topics:

- ▶ Database Access Threads
- ▶ Accumulation of DDF accounting records
- ▶ zIIP
- ▶ Using RMF to monitor distributed data

### 7.2.1 Database Access Threads

Because of Database Access Threads (DBATs)<sup>1</sup> impact on performance and accounting, the general recommendation is to use INACTIVE MODE threads instead of ACTIVE MODE threads.

Table 7-1 lists the most relevant DSNZPARM parameters affecting DBATs. Refer to Chapter 3, “Installation and configuration” on page 69 for more information about DSNZPARMs.

*Table 7-1 Summary of DSNZPARM parameters affecting DBATs*

Parameter	Description
CMTSTAT	ACTIVE or INACTIVE. It governs whether DBATs/connections remain active across commits.
MAXDBAT	Maximum number of concurrent DBATs (<=1999) or connections if CMTSTAT=ACTIVE
CONDBAT	Maximum number of concurrent connections (<=150000)
POOLINAC	POOL THREAD TIMEOUT: Specify the approximate time, in seconds that a database access thread (DBAT) can remain idle in the pool before it is terminated
IDTHTOIN	Idle thread timeout interval

The MAX USERS field on panel DSNTIPE represents the maximum number of allied threads, and the MAX REMOTE ACTIVE field on panel DSNTIPE represents the maximum number of database access threads. Together, the values you specify for these fields cannot exceed 1999.

In the MAX REMOTE CONNECTED field of panel DSNTIPE, you can specify up to 150,000 as the maximum number of concurrent remote connections that can concurrently exist within DB2. This upper limit is only obtained if you specify the recommended value INACTIVE for the DDF THREADS field of installation panel DSNTIPR.

Consider a scenario where the MAXDBAT is reached and DBATs start to queue. This can be simulated, for example, by starting many requesters in a remote server against a DB2 for z/OS where MAXDBAT was set to a convenient value of 100.

Example 7-1 on page 272 displays the DDF THD(\*) DETAIL command output of such scenario.

<sup>1</sup> DBATs are introduced in 1.7, “Connection pooling” on page 22.

#### Example 7-1 DIS THD(\*) DETAIL

```
DSNL080I  -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I  TCPRT=12347  SECPRT=12349  RESPT=12348  IPNAME=-NONE
DSNL085I  IPADDR=:9.12.6.70
DSNL086I  SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL086I  RESYNC  DOMAIN=wtsc63.itso.ibm.com
DSNL090I  DT=I  CONDBAT=    300  MDBAT=   100
DSNL092I  ADBAT=   100  QUEDBAT=   695  INADBAT=    0  CONQUED=    28
DSNL093I  DSCDBAT=    0  INACONN=    3
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Note the following fields in Example 7-1:

► **MDBAT**

Maximum number of database access threads as determined by the MAXDBAT DSNZPARM parameter. This effectively determines the maximum number of concurrent active database access threads that could potentially be executing SQL.

► **QUEDBAT**

This value reflects a cumulative counter that is always incremented when the MDBAT limit, described above, has been reached. The QUEDBAT value is equal to the cumulative number of newly attached connections, or new work on inactive connections (see DSNL093I INACONN), or new work on inactive DBATs (see INADBAT) that had to wait for a DBAT to become available to service the new work. This value is identical to the QDSTQDBT statistical value and a non-zero value suggests that performance and throughput may have been affected. See ADBAT and DSNL090I MDBAT for additional information. Also note that the QUEDBAT counter is only reset at restart for this DB2 subsystem.

So MAXDBAT has been reached and DBATs start to queue. Example 7-2 shows an OMEGAMON XE for DB2 Performance Expert on z/OS (OMEGAMON PE) Statistics Trace report short created using the SMF records cuts during this scenario. Note the field DBAT QUEUED. If the limit set by MAXDBAT parameter is reached, remote SQL requests are queued until a DBAT can be created. The number of times queuing occurred is shown by the field DBAT QUEUED.

#### Example 7-2 OM/PE Statistics Report Short showing DBAT QUEUED-MAXIMUM ACTIVE > 0

1	LOCATION: DB9A GROUP: N/P MEMBER: N/P SUBSYSTEM: DB9A DB2 VERSION: V9	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4) STATISTICS TRACE - SHORT	PAGE: 3-62 REQUESTED FROM: NOT SPECIFIED TO: NOT SPECIFIED ACTUAL FROM: 04/23/09 21:09:30.76
----- HIGHLIGHTS -----			
BEGIN RECORD: 04/23/09 21:59:30.76	TOTAL THREADS : 0	AUTH SUCC.W/OUT CATALOG: 0	DBAT QUEUED: 649
END RECORD : 04/23/09 22:04:30.76	TOTAL COMMITS : 620	BUFF.UPDT/PAGES WRITTEN: 6.51	DB2 COMMAND: 0
ELAPSED TIME: 5:00.000030	INCREMENTAL BINDS: 0	PAGES WRITTEN/WRITE I/O: 1.58	TOTAL API : 48

Example 7-3 on page 273 show an OMEGAMON PE Statistics Report Long. This report provides more detailed information.

**Example 7-3 OM/PE Statistics Report Long showing DBAT QUEUED-MAXIMUM ACTIVE > 0**

```

1  LOCATION: DB9A                      OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)          PAGE: 2-9
   GROUP: N/P                          STATISTICS REPORT - LONG                                REQUESTED FROM: ALL      00:00:01.
   MEMBER: N/P                                                                    TO: DATES      23:55:23.
   SUBSYSTEM: DB9A                                                                INTERVAL FROM: 04/27/09 14:34:28.
   DB2 VERSION: V9                                                                TO: 04/27/09 14:39:28.
                                     SCOPE: MEMBER

```

```

---- HIGHLIGHTS -----
INTERVAL START : 04/27/09 14:34:28.60  SAMPLING START: 04/27/09 14:34:28.60  TOTAL THREADS : 0.00
INTERVAL END   : 04/27/09 14:39:28.60  SAMPLING END  : 04/27/09 14:39:28.60  TOTAL COMMITS : 620.00
INTERVAL ELAPSED: 5:00.000008          OUTAGE ELAPSED: 0.000000          DATA SHARING MEMBER: N/A

```

GLOBAL DDF ACTIVITY	QUANTITY	/SECOND	/THREAD	/COMMIT	QUERY PARALLELISM	QUANTITY	/SECOND	/THREAD	/COMM
DBAT QUEUED-MAXIMUM ACTIVE	638.00	2.13	N/C	N/A	MAX.DEGREE OF PARALLELISM	0.00	N/A	N/A	N
CONV.DEALLOC-MAX.CONNECTED	0.00	0.00	N/C	N/A	PARALLEL GROUPS EXECUTED	0.00	0.00	N/C	0.
COLD START CONNECTIONS	0.00	0.00	N/C	0.00	RAN AS PLANNED	0.00	0.00	N/C	0.
WARM START CONNECTIONS	0.00	0.00	N/C	0.00	RAN REDUCED	0.00	0.00	N/C	0.
RESYNCHRONIZATION ATTEMPTED	0.00	0.00	N/C	0.00	SEQUENTIAL-CURSOR	0.00	0.00	N/C	0.
RESYNCHRONIZATION SUCCEEDED	0.00	0.00	N/C	0.00	SEQUENTIAL-NO ESA	0.00	0.00	N/C	0.
CUR TYPE 1 INACTIVE DBATS	0.00	N/A	N/A	N/A	SEQUENTIAL-NO BUFFER	0.00	0.00	N/C	0.
TYPE 1 INACTIVE DBATS HWM	1.00	N/A	N/A	N/A	SEQUENTIAL-ENCLAVE SER.	0.00	0.00	N/C	0.
TYPE 1 CONNECTIONS TERMINAT	0.00	0.00	N/A	N/A	ONE DB2 - COORDINATOR = NO	0.00	0.00	N/C	0.
CUR TYPE 2 INACTIVE DBATS	53.00	N/A	N/A	N/A	ONE DB2 - ISOLATION LEVEL	0.00	0.00	N/C	0.
TYPE 2 INACTIVE DBATS HWM	60.00	N/A	N/A	N/A	ONE DB2 - DCL TTABLE	0.00	0.00	N/C	0.
ACC QUEUED TYPE 2 INACT THR	648.00	2.16	N/A	N/A	MEMBER SKIPPED (%)	N/C			
CUR QUEUED TYPE 2 INACT THR	50.00	N/A	N/A	N/A	REFORM PARAL-CONFIG CHANGED	0.00	0.00	N/C	0.
QUEUED TYPE 2 INACT THR HWM	52.00	N/A	N/A	N/A	REFORM PARAL-NO BUFFER	0.00	0.00	N/C	0.
CURRENT ACTIVE DBATS	100.00	N/A	N/A	N/A					
ACTIVE DBATS HWM	100.00	N/A	N/A	N/A					
TOTAL DBATS HWM	153.00	N/A	N/A	N/A					
CURRENT DBATS NOT IN USE	0.00	N/A	N/A	N/A					
DBATS NOT IN USE HWM	26.00	N/A	N/A	N/A					
DBATS CREATED	0.00	N/A	N/A	N/A					
POOL DBATS REUSED	659.00	N/A	N/A	N/A					

The Global DDF Activity section of the OMEGAMON PE Statistics Report contains, among others, the following fields of interest:

► **DBAT QUEUED-MAXIMUM ACTIVE**

The number of times a DBAT was queued because it reached the DSNZPARM maximum for active remote threads, MAXDBAT.

► **CONV.DEALLOC-MAX.CONNECTED**

Number of conversations that were deallocated because the DSNZPARM limit was reached for maximum remote connected threads, CONDBAT.

► **CUR TYPE 2 INACTIVE DBATS**

The current number of inactive type 2 DBATs

► **TYPE 2 INACTIVE DBATS HWM**

The maximum number of inactive type 2 DBATs - high-water mark.

► **ACC QUEUED TYPE 2 INACT THR**

The queued receive requests for a type 2 inactive thread and the number of requests for new connections received after the maximum number of remote DBATs was reached.

► **QUEUED TYPE 2 INACT THR HWM**

The maximum number of queued RECEIVE requests for a type 2 inactive thread and the number of requests for new connections received after the maximum number of remote DBATs was reached.

► **ACTIVE DBATS HWM**

The maximum number of active database access threads. This is a high-water mark.

► **TOTAL DBATS HWM**

The maximum number of active and inactive database access threads.

This report contains many more fields of interest. You should get to know and monitor them. For instance, if you detect more than 1% of DBAT queuing you may consider increasing the value of the MAXDBAT DSNZPARM parameter. However, for this particular parameter and other examples, you need to consider the global system impact of changing the value. For instance, when increasing a frequently hit MAXBAT, you need to evaluate the impact on storage and CPU utilization.

The IDTHTOIN (idle thread timeout) parameter represents the approximate time, in seconds, that an active server thread should be allowed to remain idle before it is cancelled. After this time is exhausted, the server thread is terminated to release resources that might affect other threads.

This usually occurs for one of following reasons:

- ▶ CMTSTAT=ACTIVE and a requester application or its user did not make a request to the DB2 server for an extended period. This can happen, for example, during a lengthy user absence. As a result, the server thread becomes susceptible to being cancelled because of the timeout value.
- ▶ CMTSTAT=INACTIVE and a requester application or its user committed one of the following:
  - Failed to commit before an extended dormant period (such as user absence)
  - Committed before an extended dormant period (such as user absence), but database resources are still held because of other existing conditions.

As a result, the server thread cannot be moved to the inactive state and becomes susceptible to being cancelled because of the timeout value.

IDTHTOIN has applicability with CMTSTAT set to INACTIVE. If an inflight DBAT has not received messages for an interval, it will be aborted. If a DBAT has been pooled, then it is not idle and timer does not apply.

Example 7-4 shows the information received in the system log when a thread is cancelled because of one of the reasons described above.

*Example 7-4 Thread cancelled because of idle thread timeout threshold reached*

---

```

DSNL027I  -D9C2 SERVER DISTRIBUTED AGENT WITH 287
          LUWID=USIBMSC.SCPDB9A.C418F1FEAD95=41
          THREAD-INFO=PAOLOR4:*:*:*
          RECEIVED ABEND=04E
          FOR REASON=00D3003B
DSNL028I  -D9C2 USIBMSC.SCPDB9A.C418F1FEAD95=41 288
          ACCESSING DATA FOR
          LOCATION ::9.12.6.70
          IPADDR  ::9.12.6.70

```

---

## 7.2.2 Accumulation of DDF accounting records

The DB2 subsystem parameter ACCUMACC controls whether and when DB2 accounting data is accumulated by the user for DDF and RRSAF threads. Given its impact on the way you consider and analyze accounting records, this parameter has special importance.

When roll up occurs, the values of some fields shown in accounting reports and traces lose their meanings because of the accumulation. Thus, these fields are marked either N/P or N/C

in reports generated by OMEGAMON PE. Table 7-2 on page 275 shows the fields impacted by roll up.

*Table 7-2 Fields affected by roll up for distributed and parallel tasks*

Field name	Field short description
QPACAANM	ACTIVITY NAME
QPACAANM_VAR	ACTIVITY NAME
QPACARNA	DB2 ENTRY/EXIT - AVG.DB2 ENTRY/EXIT
QPACASCH	SCHEMA NAME
QPACASCH_VAR	SCHEMA NAME
QPACCANM	STORED PROCEDURE EVENTS
QPACCAST	SCHED.PROCEDURE SUSP TIME
QPACCONT	CONSISTENCY TOKEN
QPACEJST	ENDING TCB CPU TIME
QPACSCB	BEGINNING STORE CLOCK TIME
QPACSCE	ENDING STORE CLOCK TIME
QPACSPNS	STORED PROCEDURE EXECUTED
QPACSQLC	SQL STATEMENTS
QPACUDNU	UDF EVENTS
QPACUDST	SCHED.UDF SUSP TIME
QTXAFLG1	RES LIMIT TYPE
QTXARLID	RLF TABLE ID
QWACARNA	DB2 ENTRY/EXIT EVENTS
QWACNID	NETWORK ID VALUE
QWACSPCP	STORED PROCEDURE TCB TIME
QWACTREE	TRIG ELAP TIME UNDER ENCLAVE
QWACTRTE	TRIG TCB TIME UNDER ENCLAVE
QXMIAP	RID LIST SUCCESSFUL
QXNSMIAP	RID LIST NOT USED-NO STORAGE

The acceptable values for ACCUMACC are as follows:

- NO
  - DB2 writes an accounting record when a DDF thread:
    - ends
    - is made inactive
    - does not go inactive because used a keepdynamic(yes) package
    - or a sign-on occurs for an RRSF thread
- n (a value from 2 to 65535)

DB2 writes an accounting record every  $n$  accounting intervals for a given user, where  $n$  is the number that you specify for ACCUMACC. The default value is 10.

A parameter value of 2 or higher causes accounting records to roll up into a single record every  $n$  occurrences of the user on the thread. Even if you specify a value between 2 and 65535, an accounting record might be written prior to the  $n$ -th accounting interval for a given user in the following cases:

- An internal storage threshold is reached for the accounting rollup blocks.
- The staleness threshold is reached. The user has not rolled data into an internal block in approximately 10 minutes and DB2 considers the data stale.

A user is identified by a concatenation of values, which is specified in the AGGREGATION FIELDS field ACCUMUID. The acceptable values for ACCUMUID range from 0 to 17. The acceptable values are listed in Table 7-3.

Table 7-3 ACCUMUID acceptable values

Value	Rollup criteria	Are X'00' or X'40' values considered for rollup?
0	End user ID, transaction name, and workstation name	Yes
1	End user ID	No
2	End user transaction name	No
3	End user workstation name	No
4	End user ID and transaction name	Yes
5	End user ID and workstation name	Yes
6	End user transaction name and workstation name	Yes
7	End user ID, transaction name, and workstation name	No
8	End user ID and transaction name	No
9	End user ID and workstation name	No
10	End user transaction name and workstation name	No
11	End user ID, application name, and workstation name	Yes
12	End user ID	Yes
13	End user application name	Yes
14	End user workstation name	Yes
15	End user ID and application name	Yes
16	End user ID and workstation name	Yes
17	End user application name and workstation name	Yes

DB2 writes individual accounting records for threads that do not meet the criteria for rollup. These values can be set by DDF threads through Server Connect and Set Client calls, and by RRSAF threads through the RRSAF SIGN, AUTH SIGNON, and CONTEXT SIGNON functions.

For example, our test system was configured with the following parameters:

- ACCUMUID=1



- ▶ ACCUMACC=NO
- ▶ CMTSTAT = INACTIVE

As ACCUMACC was set to NO, no grouping of accounting records is happening. Threads will become inactive after a successful commit or rollback. DB2 trace begins collecting this data at successful thread allocation to DB2, and writes a completed record when the thread terminates, the thread becomes inactive or when the authorization ID changes.

This scenario can cause the creation of a large number of SMF records and you could decide to implement the following parameters to lower the stress in your SMF collection and management system:

- ▶ ACCUMUID=1
- ▶ ACCUMACC=10
- ▶ CMTSTAT = INACTIVE

These settings will start the rollup of accounting information every 10 transactions, where each transaction is defined, for example, by a COMMIT. Because of ACCUMUID=1, the records will be grouped by End User ID in this example, but as shown in Table 7-3 on page 276, the grouping can also be made more granular. This table is only an extract of grouping possibilities. For full capabilities, refer to *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846. If the application has not set the 'End User ID' information, then rollup does not occur.

As these two DSNZPARMs can be changed online, changes can be introduced using the tracing parameters installation panel DSNTIPN or by editing a current copy of the installation job DSNTIJUZ, without a DB2 outage.

Example 7-5 shows how we changed the DSN6SYSP section of our current DSNTIJUZ job to activate this function.

*Example 7-5 Activating accounting rollup*

DSN6SYSP	ACCUMACC=10,	X
	ACCUMUID=1,	X

We activated the changes by issuing the SET SYSPARM RELOAD command as shown in Example 7-6. The changes are effective immediately and you can verify the new DSNZPARMs values using one of the methods described in 7.4.1, “Verification of currently active DSNZPARMs” on page 307.

*Example 7-6 SET SYSPARM RELOAD command example*

-DB9A SET SYSPARM RELOAD		
DSNZ006I	-DB9A DSNZCMD1 SUBSYS DB9A SYSTEM 147	
PARAMETERS LOAD MODULE NAME DSNZPARM IS BEING LOADED		
DSN9022I	-DB9A DSNZCMD0 'SET SYSPARM' NORMAL COMPLETION	
DSNG002I	-DB9A EDM RDS BELOW HAS AN 148	
	INITIAL SIZE	18575360
	REQUESTED SIZE	18577408
	AND AN ALLOCATED SIZE	23695360
DSNZ007I	-DB9A DSNZCMD1 SUBSYS DB9A SYSTEM 149	
PARAMETERS LOAD MODULE NAME DSNZPARM LOAD COMPLETE		

Figure 7-3 for an example of the effects of ACCUMACC on accounting records.

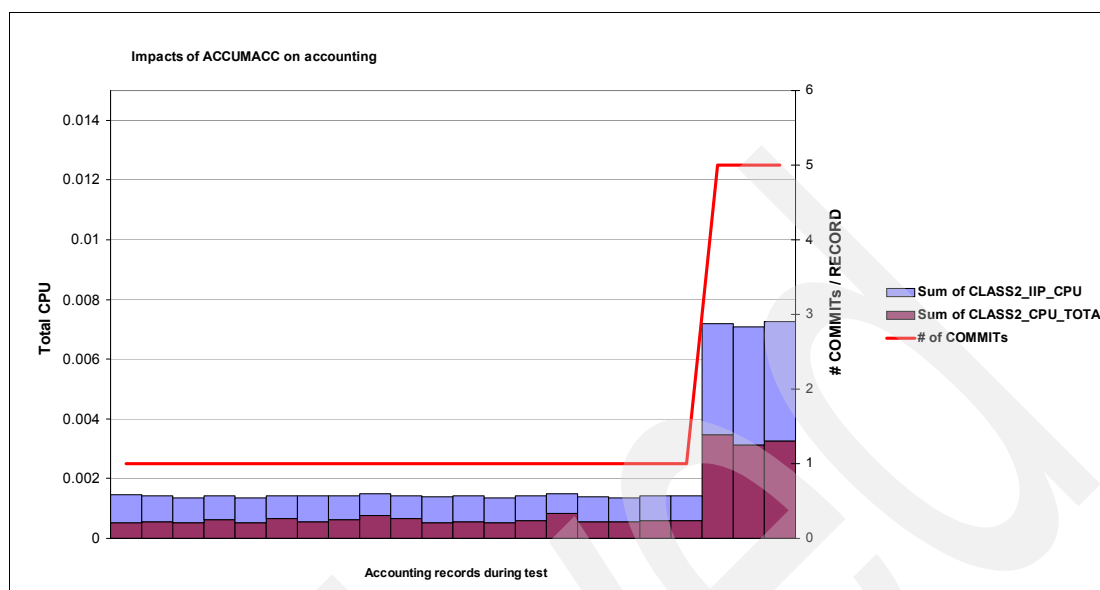


Figure 7-3 Effects of ACCUMACC on some of the fields of the accounting records

Table 7-4 shows some observed changes on fields of the table DB2PMFACCT\_DDF.

Table 7-4 ACCUMACC and effects on some accounting fields

Field	ACCUMACC=NO	ACCUMACC=n
CLIENT_ENDUSER (QWHCEUID)	Toto	Toto
CLIENT_TRANSACTION (QWHCEUTX)	TotoTestApplication	.....
PLAN_NAME (QWHCPLAN)	TotoTest	DISTSERV
MAINPACK (ADMAINPK)	SYSSH200	*ROLLUP*

As a consequence of rollup when ACCUMUID=1, you lose more granular information than the user ID. This effect depends on the level of aggregation that is defined by the parameter ACCUMUID.

For problem determination, it can be useful to deactivate the accounting rollup during a period long enough to repeat the problem and collect more granular information.

As an example, consider the accounting graph shown in Figure 7-4 on page 279. As documented in Table 7-4, the information that can help to detect the origin of the first CPU usage spike is not retained when using accounting rollup.

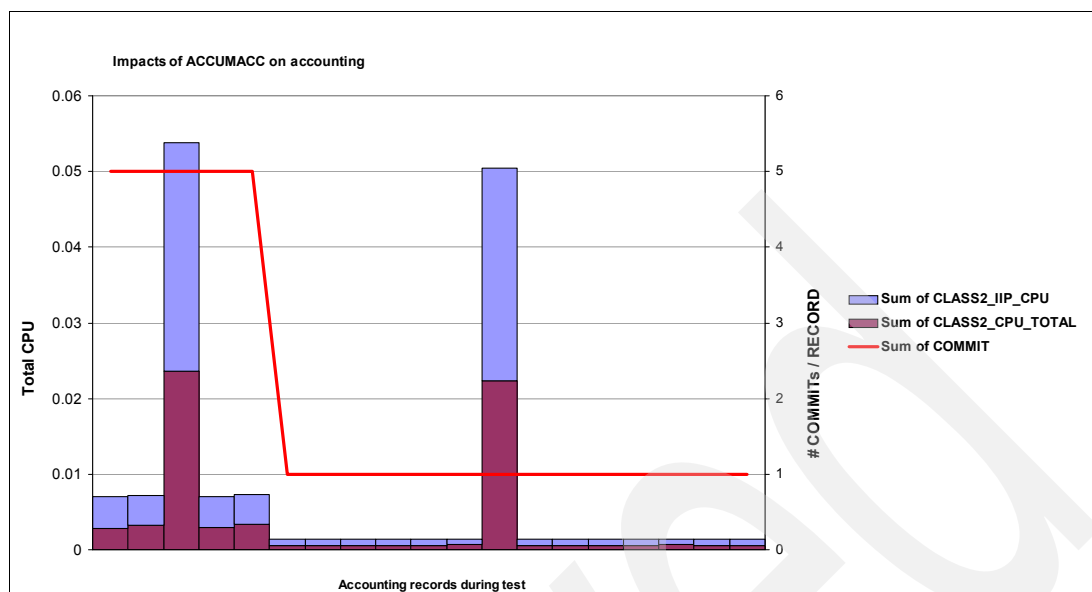


Figure 7-4 Reverting to ACCUMACC=NO

You can achieve this by changing, assembling and activating the current DSNZPARM. Example 7-7 shows how we changed the ACCUMACC parameter back to NO in our DSNTIJUZ job. The changes are made active by the execution of a SET SYSPARM RELOAD command.

Example 7-7 De-Activating accounting rollup

DSN6SYSP	ACCUMACC=NO,	X
	ACCUMUID=1,	X

## Summary

The parameter ACCUMACC indicates if rollup of accounting traces takes place and indicates the frequency of grouping records. The parameter ACCUMUID defines the aggregation fields. Both can be updated online.

Rollup of accounting information can be useful for reducing the amount of SMF data created. The summarized information created may be not adequate for problem investigation because the summarized information may hide the effects of a bad behaving thread in the rollup traces. This section showed how you can turn off accounting rollup during a problem investigation exercise. Note that if rollup is in effect, package accounting information in accounting classes 7.8 and 10 are not collected for the involved threads.

## 7.2.3 zIIP

The System z9® and System z10™ Integrated Information Processor (IBM zIIP) is a specialty engine that runs eligible database workloads. The IBM zIIP is designed to help free-up general computing capacity and lower software costs for select DB2 workloads such as business intelligence (BI), enterprise resource planning (ERP), and customer relationship management (CRM) on the mainframe.

Portions of DDF server thread processing, utility processing, and complex query parallel child processing can be directed to an IBM zIIP. The amount of general-purpose processor savings will vary based on the amount of workload executed by the zIIP, among other factors.

Refer to the IBM documentation for software and hardware requisites for zIIP at “IBM System z Integrated Information Processor (zIIP)“:

<http://www.ibm.com/systems/z/advantages/ziip/index.html>

**Important:** Other than the software and hardware pre-requisites, no special enablement procedure is necessary for DB2 to utilize zIIP. If a zIIP is installed and online, z/OS automatically manages the processing of each enclave and redirects a portion to zIIP.

The following processes can take advantage of IBM zIIP:

- ▶ DDF server threads that process SQL requests from applications that access DB2 using DRDA with TCP/IP.
- ▶ Parallel child processes. A portion of each child process executes under a dependent enclave SRB if it processes on behalf of an application that originated from an allied address space, or under an independent enclave SRB if the processing is performed on behalf of a remote application that accesses DB2 by TCP/IP. The enclave priority is inherited from the invoking allied address space for a dependent enclave or from the main DDF server thread enclave classification for an independent enclave.
- ▶ Utility index build and maintenance processes for the LOAD, REORG, and REBUILD INDEX utilities. A portion of the index build/maintenance runs under a dependent enclave SRB.
- ▶ DB2 native SQL Stored Procedures when called remotely (through DDF)

Additionally to DB2, the following components can take advantage of zIIP:

- ▶ z/OS Communications Server exploits the zIIP for eligible IPsec network encryption workloads.
- ▶ z/OS XML System Services is enabled to take additional advantage of the zIIP for eligible XML workloads.
- ▶ z/OS Global Mirror (zGM, formerly XRC Extended Remote Copy) enables DFSMS System Data Mover (SDM) processing associated with zGM/XRC to be eligible for the zIIP.
- ▶ z/OS Communications Server exploits the zIIP for select HiperSockets large message traffic.
- ▶ IBM Global Business Services can enable the Scalable Architecture for Financial Reporting (SAFR) enterprise business intelligence reporting solution for zIIP.

The following processes cannot use zIIP

- ▶ External stored procedures
- ▶ Triggers or functions that join the enclave SRB using a TCB.
- ▶ DDF server threads that use SNA to connect to DB2.

## DB2 for z/OS and sysplex routing services

Prior to z/OS V1R9.0, Work Load Manager (WLM) sysplex routing services returned the weight of a server based on the available capacity of only the regular CPU processors of a server's system adjusted by the health of the server and any work request queuing at the server.

z/OS V1R9.0 WLM Sysplex Routing services was enhanced to include an awareness of the available capacity of a server running with zIIP specialty engines. The overall weight returned for a server will now be a combined weight of the regular CPU, zIIP, and IBM System z Application Assist Processor (zAAP) capacity. This combined weight is dependent on all

servers in the sysplex running z/OS V1R9.0 or later. WLM will now also return server system weights of the regular CPU, zIIP, and zAAP available capacities.

With the exception of calling Java stored procedures, the work requested by DRDA TCP/IP clients benefits from running on server systems that have a roughly equal available capacity of regular CPU and zIIP specialty engines. Therefore, even with the z/OS V1R9.0 sysplex routing services zIIP awareness enhancements, the weighted list of servers was not optimized to favor work requests being routed to the zIIP enabled members of the DB2 data sharing group.

APAR PK38867 (DB2 V8 and 9 support for z/OS V1R9.0 sysplex routing services zIIP awareness enhancements) improves the way the server list are built: DB2 z/OS sysplex routing server list processing has been changed to take advantage of the regular CPU and zIIP weights now being returned by z/OS V1R9.0 WLM sysplex routing services. When all members of the DB2 data sharing group are running on z/OS V1R9.0, then DB2 will create a new weight for each member, which is a sum of that member's regular CPU and zIIP weights. DB2 will then re-sort the server list in descending order of these new calculated weights.

APAR PK38867 also enhances DB2 support for sysplex routing services by providing DRDA TCP/IP clients with a weighted list of servers which favors servers in the following order of precedence:

- ▶ Servers with the highest combined regular CPU and zIIP available capacity
- ▶ Servers with some combination of regular CPU and zIIP available capacity
- ▶ Servers with primarily regular CPU and minimal to no zIIP available capacity

We also recommend applying PK41236, which ensures that DB2 for z/OS sysplex routing server list always considers CPU and zIIP weights if DB2 is running on z/OS V1R9.0 or better.

## zIIP performance considerations

You can verify if zIIP processors are available to a LPAR by executing the z/OS system command `/d m=cpu`.

Example 7-8 shows an output example of this command. This kind of information can be found listed in the system log.

*Example 7-8 Output of /d m=cpu command*

---

```

RESPONSE=SC63
IEE174I 12.39.42 DISPLAY M 313
PROCESSOR STATUS
ID  CPU              SERIAL
00  +                04991E2094
01  +                04991E2094
02  +A               04991E2094
03  +A               04991E2094
04  +I               04991E2094
05  +I               04991E2094

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.00000000002991E
      Model: S18

...
A      APPLICATION ASSIST PROCESSOR (zAAP)
I      INTEGRATED INFORMATION PROCESSOR (zIIP)
...

```

---

If you do not specify any processor identifiers, the system displays the online or offline status of all processors attached to them. Whether you specify a processor identifier or not, the system displays “N” when a processor is neither online or offline, but is recognized by the machine.

**Tip:** For details about this and other commands, refer to the following Web page:

<http://publib.boulder.ibm.com/infocenter/zos/v1r10/index.jsp>

The information shown in Example 7-8 on page 281 indicates that this command was executed in a processor model 2094-710 for the IBM System z9 EC server family.

For information about the capacity of this and other System z configurations, see the following IBM Web site:

<http://www.ibm.com/systems/z/advantages/management/srm/>

**Note:** A zIIP engine always delivers its full hardware capacity even if the general purpose processors in the box are capped to run at less than full speed.

The maximum number of processor units, including zIIPs, depends on the server model. A general purpose CPU is required per each specialty engine. For details, refer to the specific System z IBM Web site, which for IBM System z9 Enterprise Class is as follows:

<http://www.ibm.com/systems/z/hardware/z9ec/specifications.html>

During the design, consider that the maximum number of zIIPs that can be made available in a server and its processing speed depend on the server Model. A zIIP processor can be shared among LPARs of the same box. Combined zAAPs and zIIPs cannot be more than twice the number of CPs.

### **zIIP related system parameters**

This section describes the system parameters for IEAOPTxx that are related to the zIIP processors, you may need to contact your local z/OS System Engineers for more details about the current settings of this parameters.

#### **PROJECTCPU**

The projected usage function (PROJECTCPU) is intended to gather information about how much CPU time is spent executing code which could potentially execute on zIIPs. This information can be gathered from production workloads or from representative workloads to understand the potential for zIIP execution with current applications even before actually owning a zIIP processor.

Setting the IEAOPTxx parmlib member option PROJECTCPU=YES directs z/OS to record the amount of work eligible for zIIP, and zAAP, processors. SMF Record Type 72 subtype 3 is input to the RMF post processor. The Workload Activity Report lists workloads by WLM service class. In this report, the field APPL% IPPCP indicates which percentage of a processor is zIIP eligible, and the field APPL% AAPCP indicates which percentage of a processor is zAAP eligible. SMF Record Type 30 provides more detail on specific address spaces.

For customers that have installed zIIPs, the reporting functions can provide current zIIP execution information that can be used for optimizing current configurations or for helping to predict potential future usage.

The acceptable values for this parameter are NO and YES. The default value is NO.

## **IIPHONORPRIORITY**

This parameter defines if zIIP eligible workload may be executed or not in a general purpose processor in cases where the zIIP is busy. The acceptable values for this parameter are YES and NO, the default is YES.

**Attention:** Standard processors can also run zIIP processor eligible work (even if IIPHONORPRIORITY is set to NO), if necessary to resolve contention for resources with non zIIP processor eligible work. It means that work that could have been executed in a zIIP processor would be executed in a standard processor if a transaction delayed by zIIPs being busy is keeping resources locked.

If the IIPHONORPRIORITY parameter is set to YES, it specifies that standard processors run both zIIP processor eligible and non-zIIP processor eligible work in priority order when the zIIP processors indicate the need for help from standard processors. The need for help is determined by the alternate wait management (AWM) function of SRM for both standard and zIIP processors. Standard processors help each other and standard processors can also help zIIP processors if YES is in effect, which is the default. Specifying YES does not mean the priorities will always be honored because the system manages dispatching priorities based on the goals provided in the WLM service definition.

If zIIP processors are defined to the LPAR but are not online, the zIIP processor eligible work units are processed by standard processors in priority order. The system ignores the IIPHONORPRIORITY parameter in this case and handles the work as though it had no eligibility to zIIP processors. The zIIP processor eligible processor times are reported in RMF and SMF for planning purposes.

IBM suggests that you specify or default to IIPHONORPRIORITY=YES.

If you set this parameter to NO, standard processors will not examine zIIP processor eligible work regardless of the demand for zIIP processors as long as there is standard processor eligible work available.

**Important:** If you work with IIPHONORPRIORITY=NO and the zIIP processors available in the system are busy, a DB2 request would wait and you may see an increase on NOT ACCOUNTED time in the DB2 accounting caused by wait on CPU (zIIP). However, on busy sub full capacity machine configurations, and as zIIPs are not capped processors, zIIP eligible workload may execute faster on a zIIP than in a general purpose processor.

## **ZIIPAWMT**

Specifies an AWM value for zIIPs to minimize SRM and LPAR low utilization effects and overhead.

In an LPAR, some n-way environments with a small workload may appear to have little capacity remaining because of the time spent waking up idle zIIPs to compete for individual pieces of work. The ZIIPAWMT parameter allows you to reduce this time so that capacity planning is more accurate and CPU overhead is reduced, even though it might take longer until arriving work gets dispatched.

The ZIIPAWMT parameters internally affects the frequency with which the specialty engine will check the need for help. If help is required, the zIIP processor signals a waiting zAAP or zIIP to help. When all zAAP or zIIP processors are busy and IIPHONORPRIORITY=YES, the zIIP processor asks for help from the standard processors. All available specialty engines must be busy before help is asked of the standard processors.

Reducing the value specified for ZIIPAWMT causes the specialty engines to request help after being busy for a shorter period of time. If IIPHONORPRIORITY is set to YES, help is provided to one CP at a time, in the priority order of zIIP processor eligible work, non-zIIP processor eligible work. Reducing the ZIIPAWMT value too low can cause the standard processors to run an excessive amount of zIIP processor eligible workload, which might result in lower priority non-zIIP processor eligible work to be delayed. Conversely, increasing the value specified for ZIIPAWMT causes the specialty engines to request help only after being busy for a longer period of time, which might delay the standard processors from providing help when it is necessary.

The acceptable values for this parameter range from 1 to 499999 microseconds, the default Value is 12000 (12 milliseconds).

## Impacts on DB2 accounting

The DB2 accounting trace records provide information related to application programs including processor resources consumed by the application.

**Important:** Accumulated zIIP CPU time is not accumulated in the existing class 1, class 2, and class 7 accounting fields.

Figure 7-5 shows an example of accounting report including the zIIP processor usage. This graph includes the zIIP CPU time as reported by the field AWACC2Z, IIP CPU TIME of OMEGAMON PE. You can expect, given the required technical conditions, about 50% of the CPU time of a DDF workload to be eligible and executed on a zIIP processor. Failing to update your reporting tools to account for the new zIIP related CPU utilization reporting fields may introduce errors in your capacity planning and cost distribution policies.

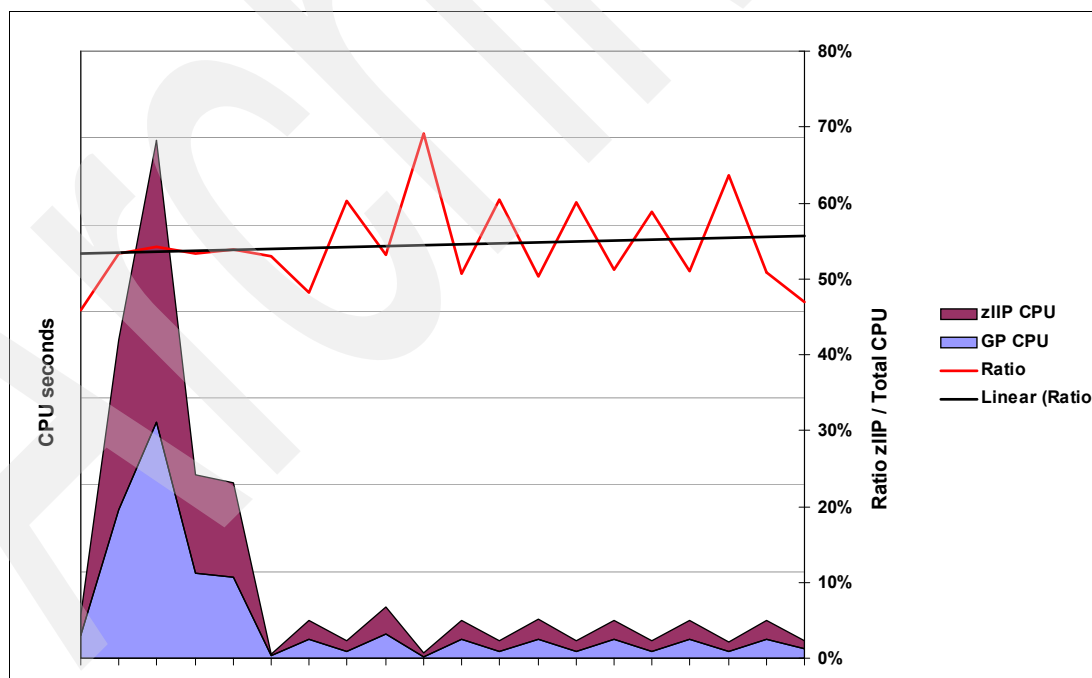


Figure 7-5 Accounting report including zIIP CPU usage

New accounting fields are defined to let users know how much time is spent on IBM zIIPs, as well as how much time zIIP eligible work overflowed to standard processors. CPU time eligible for offload to a zIIP processor is provided in the accounting records even if a zIIP is



not online or DB2 is not running on a z9 mainframe. z/OS and DB2 maintenance requirements apply.

Changes to IFCIDs 0003, 0231, 0239, 0147, and 0148 as mapped by the following macros:

- ▶ DSNDQPAC - PACKAGE/DBRM ACCOUNTING DATA MAPPING MACRO
- ▶ DSNDQWAC - ACCOUNTING DATA MAPPING MACRO
- ▶ DSNDQWHU - IFC CPU HEADER MAPPING MACRO
- ▶ DSNDQW01 - IFCID HEADER MAPPING MACRO
- ▶ DSNDQW02 - IFCID HEADER MAPPING MACRO
- ▶ DSNDQW03 - PARALLEL GROUP TASK TIME TRACE MAPPING MACRO

Example 7-9 shows some of the zIIP related fields. The field QWACZIIP\_ELIGIBLE, reporting zIIP eligible work executed on general processors, will not be populated anymore after DB2 9 for z/OS.

*Example 7-9 Extract of hlq.SDSNMACS(DSNDQWAC) macro showing zIIP related fields*

---

```

QWACZIIP EQU *
QWACCLS1_zIIP    DS  CL8 /* Accumulated CPU time consumed while */
*                /* executing on an IBM specialty engine in*/
*                /* all environments. */
QWACCLS2_zIIP    DS  CL8 /* Accumulated CPU time consumed while */
*                /* executing in DB2 on an IBM specialty */
*                /* engine. */
QWACTRTT_zIIP    DS  CL8 /* Accumulated CPU time consumed executing*/
*                /* triggers on the main application */
*                /* execution unit on an IBM specialty */
*                /* engine. */
QWACZIIP_ELIGIBLE DS  CL8 /* Accumulated CPU executed on a standard */
*                /* CP for zIIP-eligible work. */
*                /* This field will no longer be populated */
*                /* after DB2 version 9. */
QWACSPNF_zIIP    DS  CL8 /* Accumulated CPU time consumed executing*/
*                /* stored procedure requests on the main */
*                /* application execution unit on an IBM */
*                /* specialty engine. Since these SPs run */
*                /* entirely within DB2, this time */
*                /* represents class 1 and class 2 time. */
QWACUDFNF_zIIP    DS  CL8 /* ** RESERVED FOR FUTURE FUNCTION** */

```

---

**Important:** Any monitor or capacity planning program or process that you may be using needs to reflect zIIP CPU time when zIIP workload starts to be executed. It is important to account for all the CPU used by a DB2 thread when part of its processing is routed to a zIIP engine. Otherwise you may observe a drop in the total CPU usage by applications that are offloading CPU time to zIIPs.

Refer to APAR PK18454 for DB2 for z/OS V8 exploitation of the IBM System z9 Integrated Information Processor for DRDA threads.

For OMEGAMON PE V310, verify APAR PK25395: zIIP support for OMEGAMON XE for DB2 Performance Expert on z/OS V310. This APAR provides monitoring support for zIIPs processors as follows:

- ▶ Support in Reporting (Accounting, Record Trace)
- ▶ Support in all relevant user interfaces' "Thread Details" panels, for example, TEP, Classic/VTAM, and PE Client
- ▶ Performance Warehouse Support

APAR PK51045 incorporates specialty engines (zIIP and zAAP) support needed for XML data.

As an example, an OMEGAMON PE report was executed using the command shown in Example 7-10. This report shows the activity of a DRDA request executed using the CLI driver.

Example 7-10 OMEGAMON PE RECTRACE command example

```
RECTRACE
TRACE LEVEL(LONG)
INCLUDE( IFCID(003))
```

Example 7-11 shows an extract of the RECORD TRACE report.

Example 7-11 OMEGAMON PE Record Trace extract showing zIIP related fields

LOCATION: DB9A		OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)		PAGE: 1-7172	
GROUP: N/P		RECORD TRACE - LONG		REQUESTED FROM:	
....					
-----					
INSTRUMENTATION ACCOUNTING DATA					
CLASS 1 BEGINNING STORE CLOCK TIME		04/16/09 22:25:22.213037	ENDING STORE CLOCK TIME		04/16/09 22:25:27.925297
ELAPSED TIME		5.712260	MVS TCB TIME		0.05
BEGINNING MVS TCB TIME		0.000011	ENDING MVS TCB TIME		0.051343
STORED PROC ELAPSED TIME		0.000000	CONVERSION FACTOR		564
STORED PROCEDURE TCB TIME		0.000000	PAR.TASKS:		0
UDF ELAPSED TIME		0.000000	COMMITTS :		1
UDF TCB TIME		0.000000	ROLLBACKS:		0
NETWORK ID VALUE		'BLANK'	PROGRAMS :		1
REASON ACCT INVOKED:		'DDF TYPE 2 INACTIV IS BECOMING ACTIVE'			
IIP CPU TIME		0.062312			
CLASS 1/2 STORED PROC ZIIP TCB TIME		0.000000			
STORED PROC ELAPSED TIME		0.000000			
STORED PROC CP ELAPSED TIME		0.000000			
CLASS 2 DB2 ELAPSED TIME		0.558715	DB2 ENTRY/EXIT EVENTS		106
TCB TIME		0.047076	NON-ZERO CLASS 2		YES
STORED PROC ELAPSED TIME		0.000000	CLASS 2 DATA COLLECTED		YES
STORED PROCEDURE TCB TIME		0.000000	STORED PROC. ENTRY/EXITS		0
UDF ELAPSED TIME		0.000000	UDF SQL ENTRY/EXITS EVENTS		0
UDF TCB TIME		0.000000	IIP CPU TIME		0.057921
TRIG ELAP TIME UNDER ENCLAVE		0.000000	IIP ELIGIBLE CP CPU TIME		0.000000
TRIG TCB TIME UNDER ENCLAVE		0.000000	QWACTRTT_ZIIP		0.000000
TRIG ELAP TIME NOT UNDER ENCLAVE		0.000000			
...					

The following zIIP related information is indicated in bold in Example 7-11:

- ▶ **CLASS 1/2 STORED PROC ZIIP TCB TIME**

The accumulated CPU time that is consumed while running stored procedure requests on the main application execution unit on an IBM zIIP. As these stored procedures run entirely within DB2, this time represents class 1 and class 2 time

- ▶ IIP CPU TIME

The total CPU time for all executions of this package or DBRM that was consumed on an IBM zIIP

- ▶ IIP ELIGIBLE CP CPU TIME

The accumulated CPU time that ran on a standard CP for zIIP-eligible work

- ▶ QWACTRTT\_ZIIP

The accumulated CPU time consumed on an IBM specialty engine while running triggers on a nested task or on the main application execution unit.

## Monitoring zIIP utilization

To monitor the IBM specialty engine usage, you can use the following tools:

- ▶ DB2 Traces

The DB2 accounting trace records provide information related to application programs including processor resources consumed by the application. As described in this section, accumulated specialty engine CPU time is not accumulated in the existing class 1, class 2, and class 7 accounting fields.

- ▶ RMF

The Resource Measurement Facility provides information about specialty engine usage. The SMF Type 72 records contain information about specialty engine usage. Fields in SMF Type 30 records let you know how much time is spent on specialty engines, as well as how much time was spent executing specialty engine eligible work on standard processors.

Example 7-12 on page 288 shows a view of SDSF showing enclaves. You may notice the following columns which are of interest when looking at zIIP utilization:

- ▶ zIIP-Time

Cumulative zIIP time consumed by dispatchable units running in the enclave on the local system.

- ▶ zICP-Time

Cumulative zIIP on CPU time consumed by dispatchable units running in the enclave on the local system. This CPU time may have been consumed in a zIIP, but were executed in a standard processor.

- ▶ zIIP-NTime

Normalized zIIP service time, in seconds.

These columns show time consumed by dispatchable units running in the enclave on the local system. For a multisystem enclave, time consumed on other systems is not included. Refer to the IBM publication *SDSF Operation and Customization*, SA22-7670-11 for details.

### Example 7-12 SDSF view of enclaves showing zIIP utilization

Display	Filter	View	Print	Options	Help										
-----															
SDSF ENCLAVE DISPLAY			SC63	ALL	LINE 1-20 (29)										
COMMAND INPUT ==>			SCROLL ==> CSR												
PREFIX=* DEST=(ALL)			OWNER=*	SYSNAME=											
NP	NAME			SysName	SubSys	zAAP-Time	zACP-Time	zIIP-Time	zICP-Time	Promoted	zAAP-NTime	zIIP-NTime			
	340005705E	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	1.11	0.03	NO	0.00	1.11			
	3C00057066	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	1.09	0.03	NO	0.00	1.09			
	400005705F	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	1.03	0.03	NO	0.00	1.03			
	480005706F	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.84	0.02	NO	0.00	0.84			
	5000057073	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.83	0.03	NO	0.00	0.83			
	5C0005707D	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.09	0.01	NO	0.00	0.09			
	6000057070	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.45	0.01	NO	0.00	0.45			
	6800057076	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.75	0.03	NO	0.00	0.75			
	6C0005707A	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.13	0.01	NO	0.00	0.13			
	7800057043	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	1.35	0.03	NO	0.00	1.35			
	7C0005705A	01.10.00	HBB7750	SC63	DB9A	0.00	0.00	0.96	0.02	NO	0.00	0.96			

## 7.2.4 Using RMF to monitor distributed data

CPU time for enclaves generated by DDF is reported in the SMF type 30 and 72 records.

SMF type 30 records contain service for all completed or in-flight enclaves, because the previous SMF type 30 record was cut. It provides resource consumption information at the address space level. A SMF type 30 record is not granular enough to measure CPU usage by enclave. DB2 produces SMF type 101 records to provide that information. SMF type 101 is used for IFCID3, the DB2 accounting record.

SMF Type 72 records provide data collected by RMF monitor 1. There is one type 72 record for each WLM Service Class Period and WLM Report Class. It can provide you with information about the goals set for your enclaves and their actual measured values for their particular service class.

While using RMF to monitor enclaves, keep in mind the condition that an enclave is created or deleted, in conjunction with the DB2 always active or sometimes active threads.

You can use the RMF online monitor to look at enclaves, or you can use the RMF Post Processor to generate reports on SMF data on enclaves. The RMF online monitor output on enclaves is shown in Example 7-13. You can reach this panel by navigating to **3 Monitor III → 1 OVERVIEW → 6 ENCLAVE**.

### Example 7-13 RMF online monitoring of enclaves

RMF V1R10 Enclave Report				Line 1 of 23				Scroll ==> CSR			
Command ==>											
Samples: 22		System: SC63		Date: 04/23/09		Time: 14.39.39		Range: 21		Sec	
Current options:		Subsystem Type: ALL						-- CPU Util --			
		Enclave Owner:						App1%		EApp1%	
		Class/Group:						14.5		98.4	
Enclave	Attribute	CLS/GRP	P Goal	% D X	EApp1%	TCPU	USG	DLY	IDL		
*SUMMARY					26.68						
ENC00008		DDFDEF	2	20	1.806	2.456	33	62	0.0		
ENC00016		DDFDEF	2	20	1.770	2.261	17	67	0.0		
ENC00021		DDFDEF	2	20	1.756	2.317	33	58	0.0		
....											

A detailed view of the enclave can be obtained by pressing Enter when placing the cursor over an enclave. An example of the obtained panel is shown in Example 7-14.

**Example 7-14 RMF Enclave Classification Data**

---

RMF Enclave Classification Data

Details for enclave ENC00016 with token 000000BC 00056D52  
Press Enter to return to the Report panel.

- CPU Time -	-zAAP Time--	-zIIP Time--
Total 2.261	Total 0.000	Total 1.221
Delta 2.231	Delta 0.000	Delta 1.214

State	----	Using	----	-----	Delay	-----	IDL	UNK
Samples	CPU	AAP	IIP	I/O	CPU	AAP	IIP	I/O
18	0.0	0.0	5.6	11	39	0.0	28	0.0
					STO	CAP	QUE	
					0.0	0.0	0.0	0.0
								17

Classification Attributes:

More: +

Subsystem Type: DDF    Owner: DB9ADIST    System: SC63

Accounting Information . . . :

    SQL09053AIX 64BIT            db2bp            paolor4

Collection Name . . . . . : NULLID

Connection Type . . . . . : SERVER

Correlation Information . . : db2bp

LU Name . . . . . :

Netid . . . . . :

Package Name . . . . . : SQLC2G15

Plan Name . . . . . : DISTSERV

Procedure Name . . . . . :

Process Name . . . . . : db2bp

Transaction Class/Job Class:

Transaction Name/Job Name :

Userid . . . . . : PAOL0R4

Scheduling Environment . . :

Priority . . . . . :

Subsystem Collection Name :

Subsystem Instance . . . . : DB9A

Subsystem Parameter . . . :

    paolor4            kodiak.itso.ibm.

---

**Tip:** For details about RMF, refer to *Effective zSeries Performance Monitoring Using Resource Measurement Facility*, SG24-6645, and the IBM RMF Web site:

<http://www.ibm.com/servers/eserver/zseries/zos/rmf/Library/>

Example 7-15 on page 290 shows how, after sorting, you can execute a RMF CPU Activity. The options for batch reports are numerous. Refer to the RMF documentation for information about the report options that is appropriate for your needs.

### Example 7-15 RMF batch reporting

```
//PAOLO44B JOB (999,POK),REGION=5M,MSGCLASS=X,CLASS=A,
//          MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
/*JOBPARM S=SC63
/*-----
/* DRDA REDBOOK --> BASIC RMF REPORT EXAMPLE
/*-----
//POSTMFR EXEC PGM=ERBRMFPP,REGION=0M
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//MFPMMSGDS DD SYSOUT=*
//MFPIINPUT DD DISP=SHR,DSN=SMFDATA.ALLRECS.G0539V00
//SYSIN DD *
REPORTS(CPU)
/*
```

The chart in Figure 7-6 shows an excerpt of the RMF *CPU Activity Report* where the important values are highlighted.

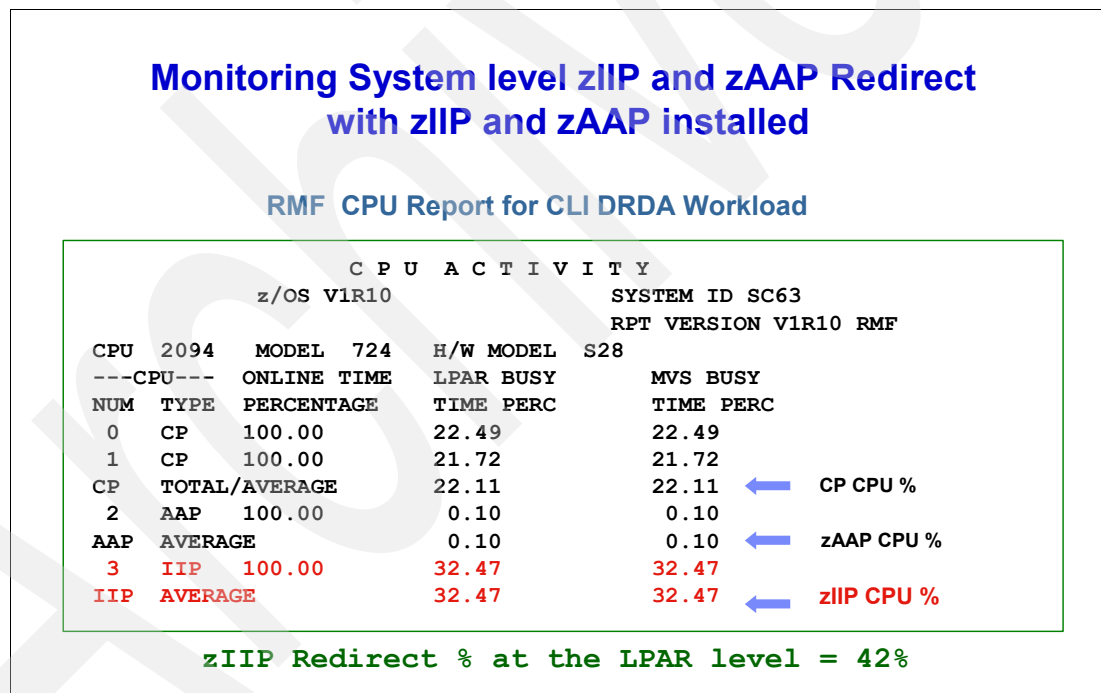


Figure 7-6 CPU report with zIIP and zAAP

The chart shows the CPU utilization of the different processors in the LPAR generated by the RMF batch CPU report.

The RMF CPU activity report shows 2 CPs, 1 zAAP engine and 1 zIIP engine.

In this case the report is for a distributed ODBC/CLI workload showing 42% zIIP redirect at the LPAR level.

This is derived by using the formula:

zIIP CP% / Total CP% (32.47 / (22.49+21.72+0.10+32.47))

The chart in Figure 7-7 shows the most important parts of a DRDA-related Workload Activity report obtained with the control card:

SYSRPTS(WLMGL(SCLASS,RCLASS,POLICY,SYSNAM(XXXX)))

In the RMF Workload Activity report look at “Service Policy” to find the zIIP redirect at the WLM Policy level. You can use the SYSNAM(XXXX) in the SYSRPTS control card to get the zIIP redirect for a specific LPAR.

We have highlighted the most important values for zIIP redirect evaluation.

► Appl% CP

Percentage of CPU time used by transactions running on standard CPs in the service or report class period.

► Appl% IIPCP

Percentage of CPU time used by zIIP eligible transactions running on standard CPs. This is a subset of APPL% CP.

► Appl% IIP

It shows the percentage of CPU time used by transactions executed on zIIPs in the service or report class period.

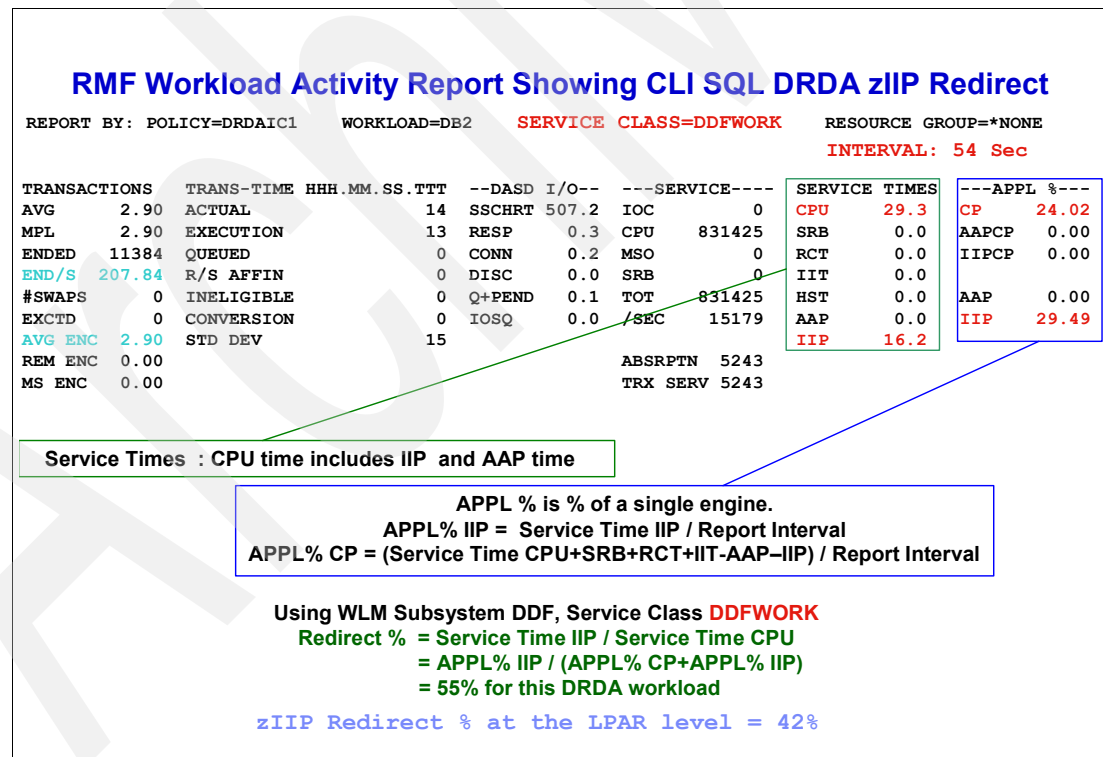


Figure 7-7 Calculating zIIP redirect %

This chart shows that a redirect percentage of 55% of the DRDA workload level using the APPL% formula.

The DRDA redirect percentage can also be calculated using the Service times:

Service Times IIP / Service Times CPU

The effective redirect percentage for this workload at the LPAR level is 42%, as shown in Figure 7-6 on page 290. It is lower at the LPAR level because of the CPU consumed by other non DB2 DRDA components such as other DB2 address spaces, TCP/IP and so forth.

### **RMF spreadsheet reporter**

The RMF Spreadsheet Reporter serves as a front-end to the RMF postprocessor on your z/OS system. With its graphics capabilities, RMF Spreadsheet Reporter allows you to analyze z/OS performance data through powerful graphical charts right from your workstation.

The main advantage of this product is ease of use. With it, you can get a clear view of the system behavior in a short period of time. This tool was extensively used during the examples in this chapter. You can get this software free of charge from the following Web page:

[http://www.ibm.com/servers/eserver/zseries/zos/rmf/tools/#spr\\_win](http://www.ibm.com/servers/eserver/zseries/zos/rmf/tools/#spr_win)

Save the RMF batch report output created, for instance, using JCL similar to the one shown in Example 7-15 on page 290, and transfer it to your workstation.

Start the software and select **Working Set** from the Create menu as shown in Figure 7-8. You need to select the Report Listing folder to make this option available.

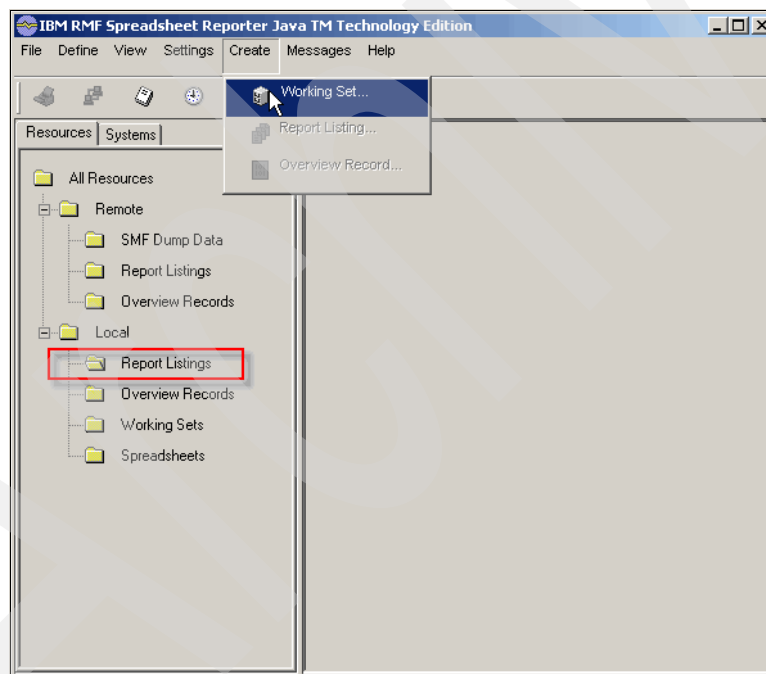


Figure 7-8 RMF Spreadsheet reporter: creating a working set menu

You will get the “Create Working Set” panel as shown in Figure 7-9 on page 293. You can provide a meaningful name to the working set under the Name section. Click **Run** to start the working set creation. This process could take some time if the report is large.



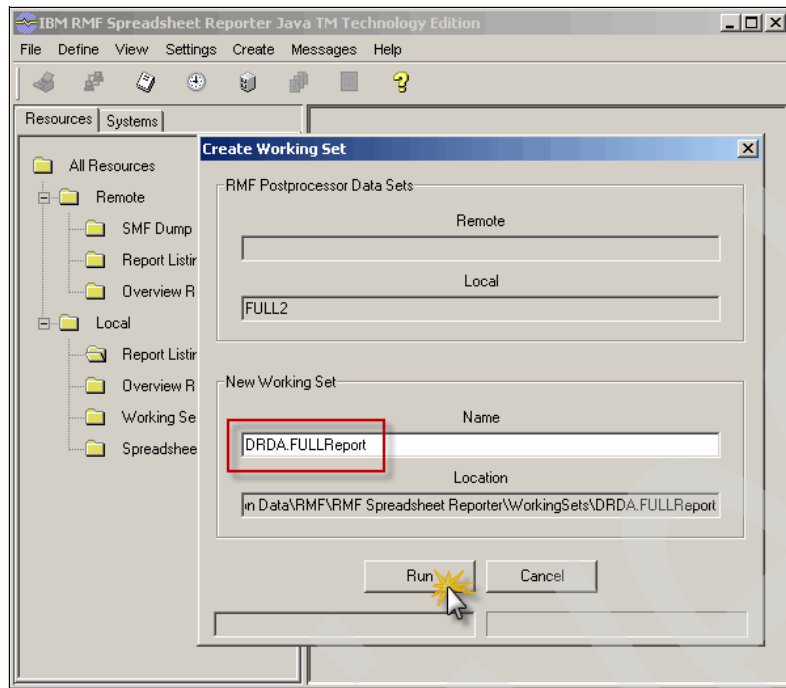


Figure 7-9 RMF Spreadsheet reporter: creating a working set

Once you get confirmation of the working set, you can go to the Spreadsheet folder and select any of the available spreadsheet reports, as shown in Figure 7-10.

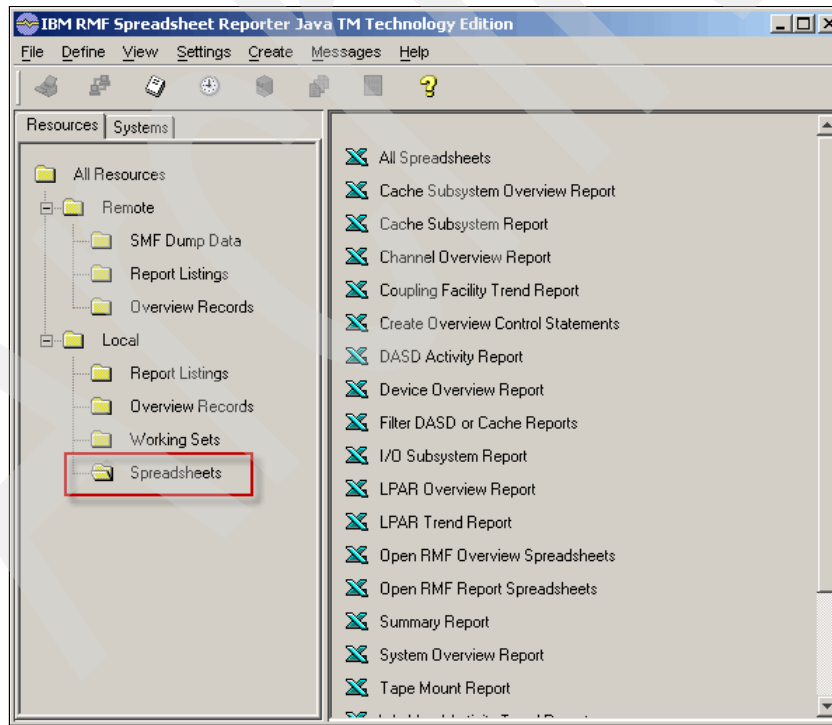


Figure 7-10 RMF Spreadsheet reporter: selecting reports

As an example of reporting, consider Figure 7-11. This report shows the Physical total dispatch time in our test LPAR during a intensive test. Note the high zIIP utilization.

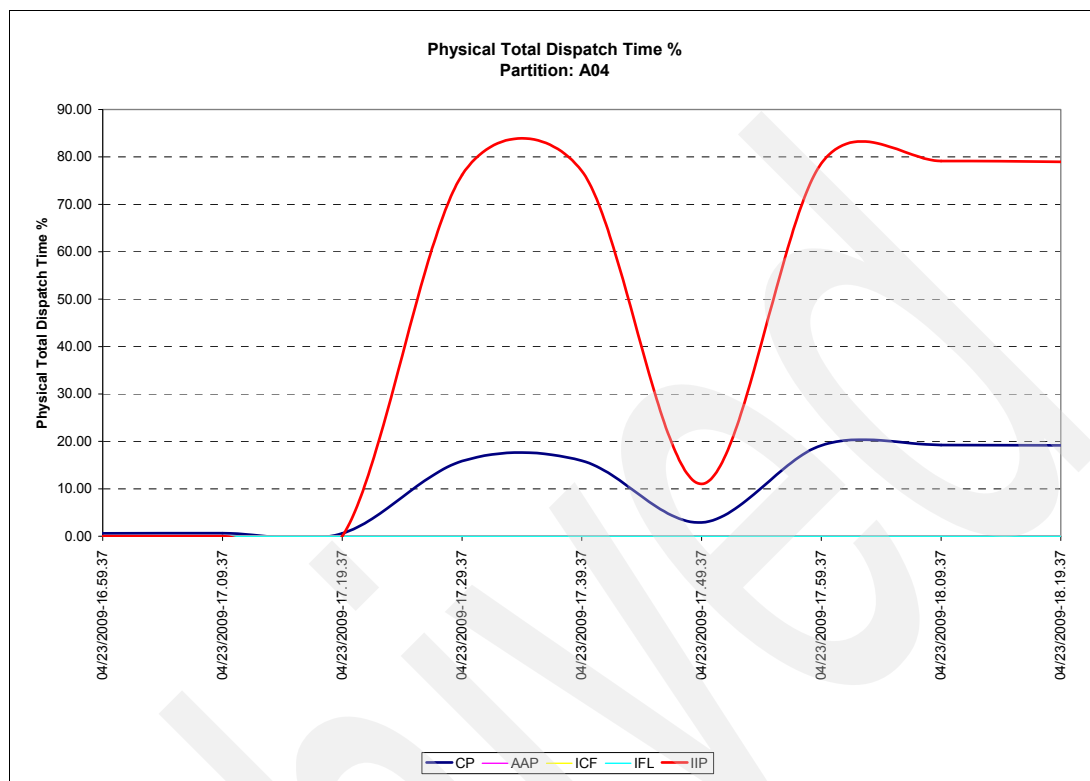


Figure 7-11 RMF report example: Physical Total Dispatch Time %

The workload reported in this example was created using the script described in Appendix D.1, “Stress tests script” on page 444.

## 7.3 Checking settings in a distributed environment

In this section we describe the following tools that are available for distributed components of a n-tier architecture to obtain configuration related information:

- ▶ db2set: DB2 profile registry command
- ▶ db2 get dbm cfg
- ▶ db2 get cli configuration
- ▶ db2pd
- ▶ Getting database connection information
- ▶ Getting online help for db2 commands
- ▶ Other useful sources of information

### 7.3.1 db2set: DB2 profile registry command

This command displays, sets, or removes DB2 for LUW profile variables. It can be issued at the DB2 client and DB2 Connect Server to obtain information related to the environment setup.

If no variable name is specified, the values of all defined variables are displayed. If a variable name is specified, only the value of that variable is displayed. To display all the defined values of a variable, specify variable -all. To display all the defined variables in all registries, specify registry -all.

To modify the value of a variable, specify variable=, followed by its new value. To set the value of a variable to NULL, specify variable = -null. Changes to settings take effect after the instance has been restarted. To delete a variable, specify variable=, followed by no value.

Example 7-16 shows the execution of this command using the -all option.

*Example 7-16 db2set -all*

---

```
$ db2set -all
[i] DB2CONNECT_IN_APP_PROCESS=NO
[i] DB2COMM=tcPIP
[g] DB2SYSTEM=kodiak.itso.ibm.com
[g] DB2INSTDEF=db2inst1
[g] DB2ADMINSERVER=dasusr1
$
```

---

Example 7-17 shows how you can use the command db2set -lr to list all the environment variables defined to DB2. This command can help you to identify its correct spelling.

*Example 7-17 db2set -lr command*

---

```
$ db2set -lr
DB2_OVERRIDE_BPF
DB2_PARALLEL_IO
DB2ACCOUNT
DB2ADMINSERVER
DB2BQTIME
....
DB2_FORCE_NLS_CACHE
DB2YIELD
DB2_AVOID_PREFETCH
DB2_COLLECT_TS_REC_INFO
DB2_GRP_LOOKUP
DB2_INDEX_FREE
DB2_MMAP_READ
...
```

---

### 7.3.2 db2 get dbm cfg

The command **db2 get dbm cfg** returns the values of individual entries in the database manager configuration file. Example 7-18 shows a partial output of this command executed in a Linux on z server.

*Example 7-18 db2 get dbm cfg output*

---

```
db2inst1@linux11:~> db2 get dbm cfg
```

```
Database Manager Configuration
```

```
Node type = Enterprise Server Edition with local and remote clients
```

```
....
```

Database manager authentication	(AUTHENTICATION) = SERVER
Cataloging allowed without authority	(CATALOG_NOAUTH) = NO
Trust all clients	(TRUST_ALLCLNTS) = YES
Trusted client authentication	(TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication	(FED_NOAUTH) = NO
....	
TCP/IP Service name	(SVCENAME) = 5000
Discovery mode	(DISCOVER) = SEARCH
Discover server instance	(DISCOVER_INST) = ENABLE
...	
No. of int. communication buffers(4KB)	(FCM_NUM_BUFFERS) = AUTOMATIC
No. of int. communication channels	(FCM_NUM_CHANNELS) = AUTOMATIC
Node connection elapse time (sec)	(CONN_ELAPSE) = 10
Max number of node connection retries	(MAX_CONNRETRIES) = 5
Max time difference between nodes (min)	(MAX_TIME_DIFF) = 60
db2start/db2stop timeout (min)	(START_STOP_TIME) = 10

---

Refer to the DB2 for LUW documentation for more details on the information provided by this command.

Example 7-19 shows how you can combine this command with the information in the UNIX /etc/services file to obtain the port on which a DB2 is listening for connections.

*Example 7-19 Getting the port on which DB2 for LUW listen*

---

```
$ db2 get dbm cfg | grep -i service
  TCP/IP Service name                (SVCENAME) = db2c_db2inst3
$ cat /etc/services | grep db2c_db2inst3
db2c_db2inst3    50002/tcp
$
```

---

### 7.3.3 db2 get cli configuration

The **GET CLI CONFIGURATION** command lists the contents of the db2cli.ini file. This command can list the entire file, or a specified section. Example 7-20 shows the syntax of this command.

*Example 7-20 GET CLI CONFIGURATION syntax*

---

```
>>-GET CLI--+-CONFIGURATION-+-+-----+----->
      +-CONFIG-----+ '-AT GLOBAL LEVEL-'
      '-CFG-----'

>--+-----+-----><
    '-FOR SECTION--section-name-'
```

---

Example 7-21 on page 297 shows an example of execution output.

Example 7-21 GET CLI CONFIGURATION output example

---

```
$ db2 get cli configuration

Section: tstcli1x
-----
uid=userid
pwd=*****
autocommit=0
TableType='TABLE','VIEW','SYSTEM TABLE'

Section: tstcli2x
-----
SchemaList='OWNER1','OWNER2',CURRENT SQLID

Section: MyVeryLongDBALIASName
-----
dbalias=dbalias3
SysSchema=MYSHEMA
```

---

Refer to Chapter 8, “Problem determination” on page 323 for details about this command and how to update the CLI configuration using commands.

### 7.3.4 db2pd

**db2pd** is a DB2 database command available for monitoring and troubleshooting DB2 for LUW. (This command is not available for DB2 for z/OS). It is often used for troubleshooting because it can return immediate information from the DB2 memory sets.

**db2pd** provides more than twenty options to display information about database transactions, table spaces, table statistics, dynamic SQL, database configurations, and many other database details. For details, see the following Web page:

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0504poon2/>

Example 7-22 is an extract of the **db2pd** syntax showing the **-sysplex** option. This option returns information about the list of servers associated with the database alias indicated by the **db** parameter. If the **-database** parameter is not specified, information is returned for all databases.

Example 7-22 db2pd sysplex syntax

---

```
>>-db2pd--+-+-----+-----+-----+-----+----->
          '- -inst-' '- -help-' '- -version-'
>--+-+-----+-----+-----+-----+----->
      '- -sysplex--+-+-----+-----+-----+-----'
                  '-db=database-' '-file=filename-'
```

---

Example 7-23 on page 298 shows the output of the command **db2pd -sysplex**. Without the **db2pd -sysplex** command, the only way to report the sysplex list is through a DB2 trace. In our working environment, DB9A is a single system and DB9C is a Data Sharing Group.

*Example 7-23 db2pd usage example: Getting the server list*

```
$ db2pd -sysplex db=DB9A

Database Partition 0 -- Active -- Up 14 days 02:19:11

Sysplex List:
Alias:          DB9A
Location Name:  DB9A
Count:          1

IP Address      Port      Priority  Connections Status  PRDID
9.12.6.70       12347    65535    2           0      DSN09015
$ db2pd -sysplex db=DB9C

Database Partition 0 -- Active -- Up 14 days 02:19:23

Sysplex List:
Alias:          DB9C
Location Name:  DB9C
Count:          3

IP Address      Port      Priority  Connections Status  PRDID
9.12.6.70       38320    53        0           0
9.12.4.202      38320    53        0           0
9.12.6.9        38320    21        0           0
```

### 7.3.5 Getting database connection information

DB2 Connect uses the following directories to manage database connection information:

- ▶ The system database directory, which contains name, node, and authentication information for every database that DB2 Connect accesses

Use the **list db directory** command to view the information associated with the databases cataloged in your system. Example 7-24 shows the syntax of this command.

*Example 7-24 Syntax of the list database directory command*

```
>>-LIST--+-DATABASE-+-DIRECTORY-+-+-----+-----><
          '-DB-----'          '-ON--+-path--+-'
                                   '-drive-'
```

Example 7-25 shows a portion of its execution output.

*Example 7-25 db2 list db directory command output example*

```
$ db2 list db directory

System Database Directory

Number of entries in the directory = 7

Database 1 entry:

Database alias          = DB9AS
Database name           = DB9AS
Node name               = SSLNODE
Database release level  = c.00
Comment                 =
```

Directory entry type	= Remote
Authentication	= SERVER
Catalog database partition number	= -1
Alternate server hostname	=
Alternate server port number	=

Database 2 entry:

Database alias	= DB9AS2
Database name	= DB9AS2
Node name	= SSLNODE
Database release level	= c.00
Comment	=
Directory entry type	= Remote
Authentication	= SERVER_ENCRYPT
Catalog database partition number	= -1
Alternate server hostname	=
Alternate server port number	=

Database 3 entry:

....

---

The most important fields are:

- Database alias

The value of the alias parameter when the database was created or cataloged. If an alias was not entered when the database was cataloged, the database manager uses the value of the database name parameter when the database was cataloged. This is the value used in an application when connecting to a database.

- Database name

The value of the database name parameter when the database was cataloged. This name is usually the name under which the database was created.

- Node name

The name of the remote node. This name corresponds to the value entered for the nodename parameter when the database and the node were cataloged.

- Directory entry type

This value indicates the location of the database. A *Remote* entry describes a database that resides on another node.

- Authentication

The authentication type cataloged at the client

- The node directory, which contains network address and communication protocol information for every host or System z database server that DB2 Connect accesses.

The node directory is created and maintained on each database client. The directory contains an entry for each remote workstation having one or more databases that the client can access. The DB2 client uses the communication endpoint information in the node directory whenever a database connection or instance attachment is requested.

The entries in the directory also contain information about the type of communication protocol to be used to communicate from the client to the remote database partition.

Use the **list node directory** command for listing the contents of the node directory. The syntax of this command is shown in Example 7-26 on page 300.

---

*Example 7-26 Syntax of the list node directory command*

---

```
>>-LIST---+-----+---NODE DIRECTORY---+-----+-----><
          '-ADMIN-'                      '-SHOW DETAIL-'
```

---

Example 7-27 shows a partial output of this command.

---

*Example 7-27 list node directory command output*

---

```
$ db2 list node directory
```

Node Directory

Number of entries in the directory = 4

Node 1 entry:

Node name	= SC63TS
Comment	=
Directory entry type	= LOCAL
Protocol	= TCPIP
Hostname	= wtsc63.itso.ibm.com
Service name	= 12347

Node 2 entry:

Node name	= SSLNODE
Comment	=
Directory entry type	= LOCAL
Protocol	= TCPIP
Hostname	= wtsc63.itso.ibm.com
Service name	= 12349
Security type	= SSL

Node 3 entry:

...

---

Some important fields:

- Node name

The name of the remote node. This corresponds to the name entered for the `nodename` parameter when the node was cataloged.

- Directory entry type

LOCAL means the entry is found in the local node directory file. LDAP means the entry is found on the LDAP server or LDAP cache

- Protocol: Indicates the communications protocol cataloged for the node.

- The database connection services (DCS) directory, which contains information specific to host or System i database server databases.

Use the **list dcs directory** command to list the contents of the DCS. Example 7-28 shows the syntax of this command.

---

*Example 7-28 Syntax of the list dcs directory command*

---

```
>>-LIST DCS DIRECTORY-----><
```

---



Example 7-29 shows partially the output of this command.

*Example 7-29 list dcs directory command output example*

---

```
$ db2 list dcs directory
```

Database Connection Services (DCS) Directory

Number of entries in the directory = 6

DCS 1 entry:

Local database name	= DB9A
Target database name	=
Application requestor name	=
DCS parameters	=
Comment	=
DCS directory release level	= 0x0100

DCS 2 entry:

Local database name	= DB9AS
Target database name	= DB9A
Application requestor name	=
DCS parameters	=
Comment	=
DCS directory release level	= 0x0100

DCS 3 entry:

....

---

Some important fields:

- Local database name

Specifies the local alias of the target host database. This corresponds to the database name parameter entered when the host database was cataloged in the DCS directory.

- Target database name

Specifies the name of the host database. This corresponds to the target database name parameter entered when the host database was cataloged in the DCS directory.

- DCS parameters

String that contains the connection and operating environment parameters to use with the application requester. Corresponds to the parameter string entered when the host database was cataloged. This is where the sysplex support is configured. The string must be enclosed by double quotation marks, and the parameters must be separated by commas.

### 7.3.6 Getting online help for db2 commands

In case you do not remember all the valid options of a particular DB2 for LUW command, all Command Line Processor (CLP) commands can invoke a help panel at the CLP prompt by preceding the command keyword with a question mark. For many of the system commands, a summarizing help panel can be displayed by issuing the command keyword followed by a help option.

To display a CLP command help panel, preface the command keyword with a question mark at the db2 interactive mode prompt. Example 7-30 shows an extract of the output of the command **db2 ? get** executed in the CLP.

*Example 7-30 Getting online help using the CLP*

---

```
C:\Program Files\IBM\SQLLIB\BIN>db2 ? get
GET ADMIN CONFIGURATION [FOR NODE node-name [USER username USING password]]
...
GET DATABASE CONFIGURATION [FOR database-alias] [SHOW DETAIL]

GET DATABASE MANAGER CONFIGURATION [SHOW DETAIL]

GET DATABASE MANAGER MONITOR SWITCHES
[AT DBPARTITIONNUM db-partition-number | GLOBAL]
...
```

---

Most of the system commands can display a command help panel by entering the system command keyword followed by a help option. Many system commands use a common help option, while other system commands may use different and additional help options. For the first attempts, without having to search for a command's forgotten help option, try the following most common options which are likely to result in successfully invoking the command help panel:

- ▶ -h
- ▶ -?
- ▶ -help
- ▶ nothing entered after the command keyword. This may cause some commands to be actually executed.

Example 7-31 shows an output extract of using the -h option with the command **db2pd**.

*Example 7-31 Using the -h option with a system command*

---

```
C:\Program Files\IBM\SQLLIB\BIN>db2pd -h

Usage:
  -h | -help [file=<filename>]
      Help
  -v | -version [file=<filename>]
      Version
  -osinfo [disk] [file=<filename>]
      Operating System Information
  -dbpartitionnum <num>[,<num>]
      Database Partition Number(s)
  -alldbpartitionnums
      All partition numbers
  -database | -db <database>[,<database>]
      Database(s)
  -alldatabases | -alldbs
      All Active Databases
  -inst
      Instance scope output
  -file <filename>
      All Output to Filename
  -command <filename>
      Read in predefined options
```

- interactive  
Interactive
- full  
Expand output to full length
- repeat [num sec] [count]  
Repeat every num seconds (default 5) count times
- everything  
All options on all database partitions

---

You can always refer to the IBM DB2 Database for Linux, UNIX, and Windows Information Center for more detailed information. It can be found, for Version 9.5, at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>

### 7.3.7 Other useful sources of information

Example 7-32 shows how you can use the **db2licm** command to get licence information from a DB2 Connect or DB2 Enterprise Server Edition (ESE) server. You need to be logged in as the instance owner or have sourced the db2profile profile.

*Example 7-32 Using the command db2licm*

---

```
$ db2licm -l
Product name:          "DB2 Enterprise Server Edition"
License type:          "Trial"
Expiry date:           "06/21/2009"
Product identifier:     "db2ese"
Version information:    "9.5"

Product name:          "DB2 Connect Server"
License type:          "Trial"
Expiry date:           "06/21/2009"
Product identifier:     "db2consv"
Version information:    "9.5"
```

---

Example 7-33 shows how you can get software level information, including the production version and FixPack applied.

*Example 7-33 Using the command db2level*

---

```
$ db2level
DB21085I  Instance "db2inst3" uses "64" bits and DB2 code release "SQL09053"
with level identifier "06040107".
Informational tokens are "DB2 v9.5.0.3", "s081118", "U818975", and FixPack
"3".
Product is installed at "/opt/IBM/db2/V9.5_ESE".
```

---

To get the port on which the DB2 Server is listening, refer to Example 7-34 on page 304. This will provide the service that is defined (UNIX) in the file /etc/services.

---

*Example 7-34 Getting the service associated with the DB2 Server*

---

```
$ db2 get dbm cfg | grep -i service
TCP/IP Service name                (SVCENAME) = db2c_db2inst3
$
```

---

Example 7-35 shows how to get the port information from the `/etc/services` file.

---

*Example 7-35 Getting the port number from /etc/services*

---

```
$ cat /etc/services | grep -i db2c_db2inst3
db2c_db2inst3    50002/tcp
```

---

For getting the IP address of a target server when you know its name (dns entry), see Example 7-36.

---

*Example 7-36 Getting the IP address of a DB2 Connect or ESE server from its dns entry*

---

```
C:\>ping kodiak.itso.ibm.com
```

Pinging kodiak.itso.ibm.com [9.12.5.149] with 32 bytes of data:

```
Reply from 9.12.5.149: bytes=32 time=89ms TTL=128
Reply from 9.12.5.149: bytes=32 time=88ms TTL=128
Reply from 9.12.5.149: bytes=32 time=90ms TTL=128
Reply from 9.12.5.149: bytes=32 time=86ms TTL=128
```

Ping statistics for 9.12.5.149:

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 86ms, Maximum = 90ms, Average = 88ms
```

---

For getting your own IP address, see Example 7-37.

---

*Example 7-37 Getting your IP address in a windows machine*

---

```
C:\>ipconfig
```

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

```
Connection-specific DNS Suffix  . : localdomain
IP Address. . . . .               : 172.16.209.128
Subnet Mask . . . . .             : 255.255.255.0
Default Gateway . . . . .         : 172.16.209.2
```

---

See Example 7-38 if you work with an AIX server.

*Example 7-38 Getting the IP address of an AIX server*

---

```
$ ifconfig -l
en0 lo0
$ ifconfig en0
en0:
flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,
CHECKSUM_OFFLOAD(ACTIVE),PSEG,LARGESEND,CHAIN>
    inet 9.12.5.149 netmask 0xfffffc00 broadcast 9.12.7.255
    tcp_sendspace 131072 tcp_recvspace 65536 rfc1323 0
```

---

You may need to get the DNS entry for an IP address. For instance, you find the message of Example 7-39 in the DB2 SYSPRINT.

*Example 7-39 DB2 SYSPRINT*

---

```
14.06.14 STC27893 DSNL032I -DB9A DSNLIRTR DRDA EXCEPTION CONDITION IN 382
382 REQUEST FROM REQUESTOR LOCATION=::9.30.28.156 FOR THREAD WITH
382 LUWID=G90C0646.J03D.C3F9275F8EA7
382 REASON=00D3101A
382 ERROR ID=DSNLIRTR0003
382 IFCID=0192
382 SEE TRACE RECORD WITH IFCID SEQUENCE NUMBER=0000000E
```

---

You may need the server name related to the IP address for problem determination. On a Windows system, you can use the command **ping -a** followed by the IP address to resolve its host name, as shown in Example 7-40.

*Example 7-40 ping -a example command for resolving a host name*

---

```
C:\Documents and Settings\VRes01>ping -a 9.30.28.156
```

```
Pinging LENOVO-B6AFDE0A-009030028156.sv1.ibm.com [9.30.28.156] with 32 bytes of data:
```

```
Reply from 9.30.28.156: bytes=32 time=3ms TTL=128
Reply from 9.30.28.156: bytes=32 time<1ms TTL=128
Reply from 9.30.28.156: bytes=32 time<1ms TTL=128
Reply from 9.30.28.156: bytes=32 time=1ms TTL=128
```

```
Ping statistics for 9.30.28.156:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

---

When you encounter a problem, you can use the following diagnostics tool to collect relevant information for problem analysis:

- ▶ All diagnostic data including dump files, trap files, error logs, notification files, and alert logs are found in the path specified by the diagnostic data directory path (diagpath) database manager configuration parameter:

If the value for this configuration parameter is null, the diagnostic data is written to one of the following directories or folders:

- For Linux and UNIX environments: INSTHOME/sqllib/db2dump, where INSTHOME is the home directory of the instance.
- For supported Windows environments:
  - If the DB2INSTPROF environment variable is not set then x:\SQLLIB\DB2INSTANCE is used where x:\SQLLIB is the drive reference and the directory specified in the DB2PATH registry variable, and the value of DB2INSTANCE has the name of the instance.
  - If the DB2INSTPROF environment variable is set then x:\DB2INSTPROF\DB2INSTANCE is used where DB2INSTPROF is the name of the instance profile directory and DB2INSTANCE is the name of the instance (by default, the value of DB2INSTDEF on Windows 32-bit operating systems).
- ▶ For Windows operating systems, you can use the Event Viewer to view the administration notification log.
- ▶ The available diagnostic tools that can be used include db2trc, db2pd, and db2support. You can always refer to the IBM DB2 Database for Linux, UNIX, and Windows Information Center for more detailed information. It can be found, for version 9.5, at the following Web page:  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>
- ▶ For Linux and UNIX operating systems, you can use the ps command, which returns process status information about active processes to standard output.
- ▶ For UNIX operating systems, use the core file that is created in the current directory when severe errors occur. It contains a memory image of the terminated process, and can be used to determine what function caused the error.

The db2 database manager configuration parameter diaglevel specifies the type of diagnostic errors that will be recorded in the db2diag.log file. This online configurable parameter applies to the following areas:

- ▶ Database server with local and remote clients
- ▶ Client
- ▶ Database server with local clients
- ▶ Partitioned database server with local and remote clients

Valid values for this parameter are as follows:

- ▶ 0: No diagnostic data captured
- ▶ 1: Severe errors only
- ▶ 2: All errors
- ▶ 3: All errors and warnings
- ▶ 4: All errors, warnings and informational messages

The default value is 3, all errors and warnings are reported. You may need to change this parameter to 4 during problem determination. This parameter can be changed using commands shown in Example 7-41.

*Example 7-41 Updating db2 diaglevel*

---

```
db2 update dbm cfg using diaglevel 4
```

---

## 7.4 Obtaining information about the host configuration

There are several places where you can find host configuration information that affect distributed workloads or that shows the current status. The most relevant sources of information are described briefly in this section:

- ▶ Verification of currently active DSNZPARMs
- ▶ SYSPLAN and SYSPACKAGES
- ▶ Example 7.4.3 on page 310
- ▶ GET\_CONFIG and GET\_SYSTEM\_INFO stored procedures
- ▶ DB2 commands

### 7.4.1 Verification of currently active DSNZPARMs

You can use the following methods, for example, for getting the current DSNZPARMs values in effect:

- ▶ Optimization Service Center subsystem parameter panel
- ▶ The IBM provided DSNWZP stored procedure
- ▶ OMEGAMON PE batch reporting

You can specify the ddname SYSPRMDD to get a System Parameter report. This ddname is optional and applies to all the OMEGAMON PE reports.

Example 7-42 shows an extract of a sample report showing some of the DSNZPARM parameters of interest for this chapter.

*Example 7-42 Extract of and OMEGAMON PE System Parameter Report*

---

LOCATION: DB9A	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)
GROUP: N/P	SYSTEM PARAMETERS REPORT
...	
STORAGE SIZES INSTALLATION PARAMETERS (DSNTIPC,DSNTIPE)	IRLM START PROCEDURE NAME (IRLMPRC).....DB9AIRLM
-----	SECONDS DB2 WILL WAIT FOR IRLM START (IRLMSWT).....300
MAX NO OF USERS CONCURRENTLY RUNNING IN DB2 (CTHREAD).....200	U LOCK FOR REPEATABLE READ OR READ STABILITY (RRULOCK).....NO
MAX NO OF TSO CONNECTIONS (IDFORE).....50	X LOCK FOR SEARCHED UPDATE/DELETE (XLKUPDLT).....NO
MAX NO OF BATCH CONNECTIONS (IDBACK).....50	IMS/BMP TIMEOUT FACTOR (BMPTOUT).....4
MAX NO OF REMOTE CONNECTIONS (CONDBAT).....300	IMS/DLI TIMEOUT FACTOR (DLITOUT).....6
MAX NO OF CONCURRENT REMOTE ACTIVE CONNECTIONS (MAXDBAT)....100	WAIT FOR RETAINED LOCKS (RETLWAIT).....0
...	
TRACING, CHECKPOINT & PSEUDO-CLOSE PARAMETERS (DSNTIPN)	COPY2 ARCHIVE LOG DEVICE TYPE (UNIT2).....'BLANK'
-----	SPACE ALLOCATION METHOD (ALCUNIT).....BLOCK
...	
DDF/RRSAF ACCUM (ACCUMACC).....0	QUIESCE PERIOD (QUIESCE).....5
AGGREGATION FIELDS (ACCUMUID).....0	SINGLE VOLUME (SVOLARC).....NO
...	
DISTRIBUTED DATA FACILITY PANEL 2 (DSNTIP5)	DEFINE GROUP OR MEMBER (DSNTIPK)
-----	-----
TCP/IP ALREADY VERIFIED (TCPALVER).....NO	GROUP NAME (GRPNAME).....DSNCA
TCP/IP KEEPALIVE (TCPKPALV).....120	MEMBER NAME (MEMBNAME).....DSN1
EXTRA BLOCKS REQ (EXTRAREQ).....100	PARALLELISM ASSISTANT (ASSIST).....NO
EXTRA BLOCKS SRV (EXTRASRV).....100	PARALLELISM COORDINATOR (COORDNTR).....NO
POOL THREAD TIMEOUT (POOLINAC).....120	
DATABASE PROTOCOL (DBPROTCL).....N/A	
HOP SITE AUTHORIZATION (HOPAUTH).....PKGOWNER	
...	
PROTECTION INSTALLATION PARAMETERS (DSNTIPP)	DISTRIBUTED DATA FACILITY PANEL 1 (DSNTIPR)
-----	-----
DB2 AUTHORIZATION ENABLED (AUTH).....YES	DDF STARTUP OPTION (DDF).....AUTO
SYSTEM ADMINISTRATOR 1 AUTHORIZATION ID (SYSADM).....PAOLOR9	RLST ACCESS ERROR (RLFERRD).....NOLIMIT
SYSTEM ADMINISTRATOR 2 AUTHORIZATION ID (SYSADM2).....HAIMO	RESYNCHRONIZATION INTERVAL IN MINUTES (RESYNC).....2
SYSTEM OPERATOR 1 AUTHORIZATION ID (SYSOPR1).....SYSOPR	DBAT STATUS (CMTSTAT).....INACTIVE
SYSTEM OPERATOR 2 AUTHORIZATION ID (SYSOPR2).....SYSOPR	IDLE THREAD TIMEOUT INTERVAL (IDTHTOIN).....120
	EXTENDED SECURITY (EXTSEC).....YES
	MAX TYPE 1 INACTIVE THREADS (MAXTYPE1).....0

---

OMEGAMON PE also provides a view of the DSNZPARM parameters using the GUI interface, as shown in Figure 7-12.

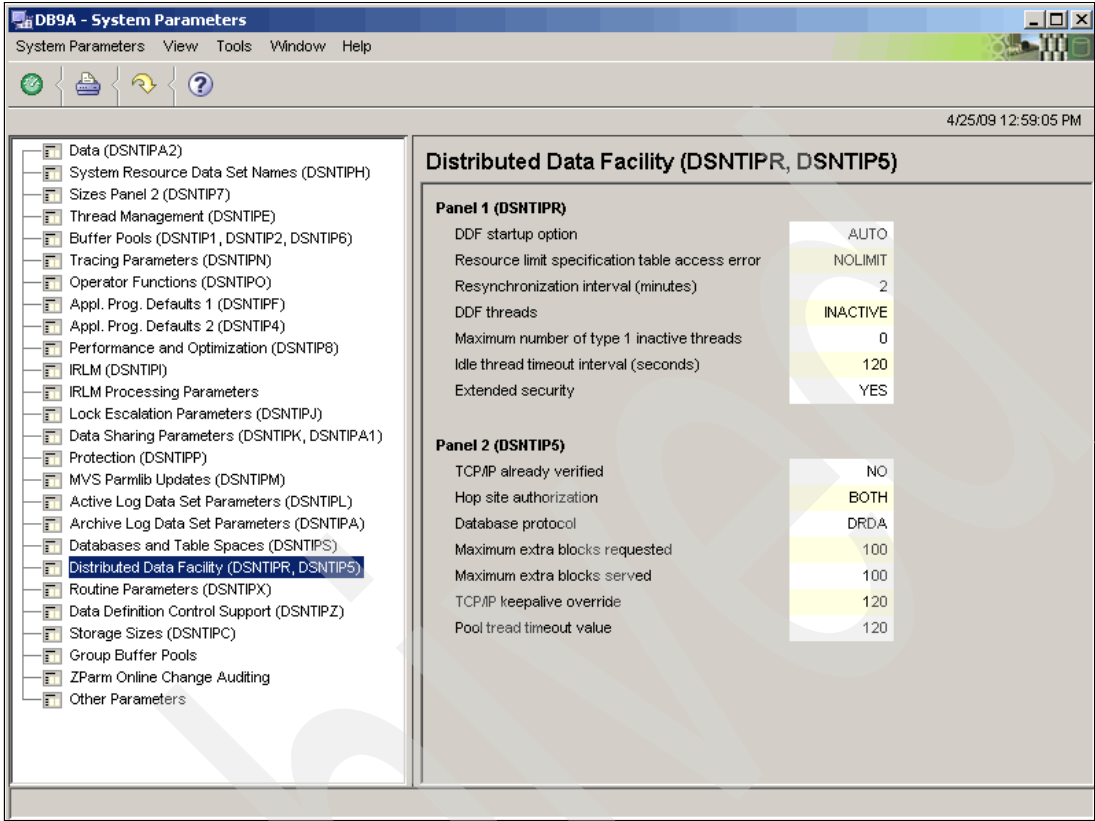


Figure 7-12 DB2 PE System parameters GUI view

Figure 7-13 on page 309 shows a panel shot of the Optimization Service Center Subsystem parameter panel. Refer to the following Web page for more information about OSC and instructions for getting a copy free of charge:

<http://www.ibm.com/software/data/db2/zos/downloads/osc.html>



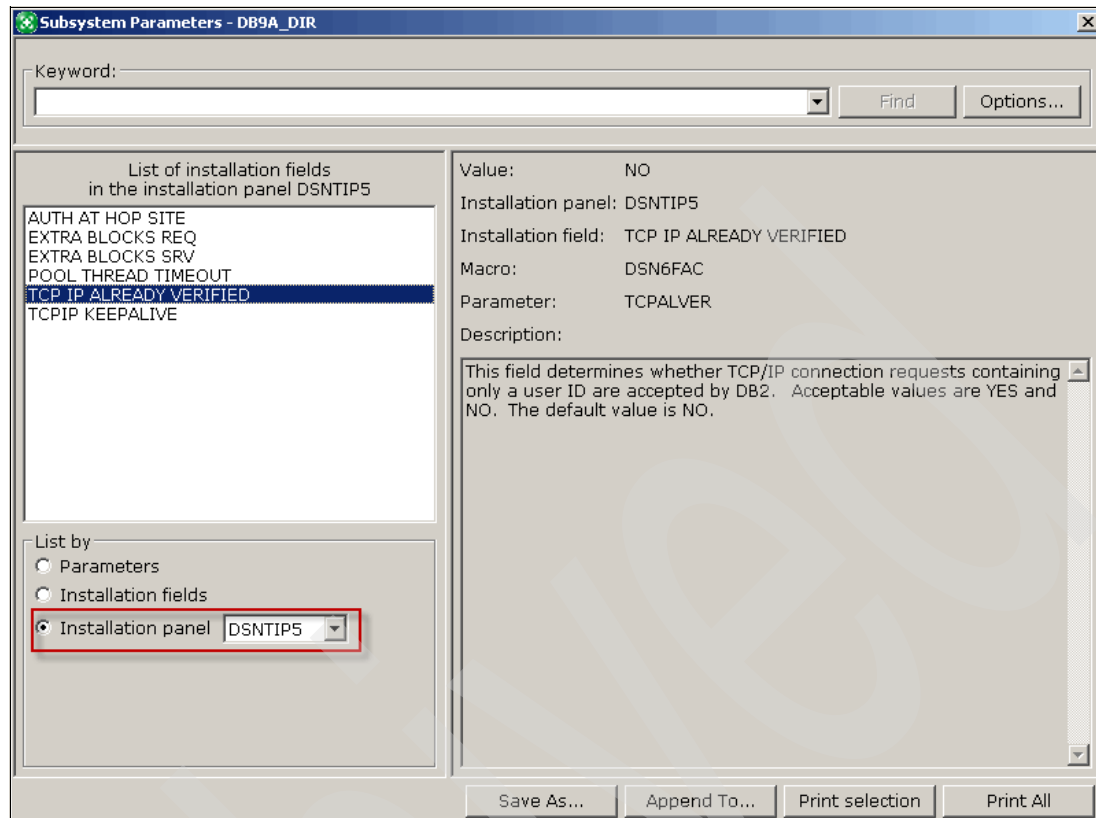


Figure 7-13 OSC Panel DSNZPARMs

## 7.4.2 SYSPLAN and SYSPACKAGES

The SYSIBM.SYSPLAN catalog table contains one row for each application plan, and the SYSIBM.SYSPACKAGE contains a row for every package.

Table 7-5 lists some of the columns that provide information that affects distributed workloads. You may need to check them during problem investigation or when migrating from Private Protocol to DRDA. Some of the column values are set using BIND or REBIND options. The migration from PP to DRDA and the BIND options are discussed in Chapter 5, “Application programming” on page 185.

Table 7-5 *SYSPLAN columns of interest for distributed workloads*

Column	Description
DBPROTOCOL	Whether remote access for SQL with three-part names is implemented with DRDA or DB2 private protocol access. P=DB2 private protocol, D=DRDA
CURRENTSERVER	Location name specified with the CURRENTSERVER option when the plan was last bound. Blank if none was specified, implying that the first server is the local DB2 subsystem.

For several reasons, you may be better off working using packages than plans for your applications. Refer to *DB2 9 for z/OS: Packages Revisited*, SG24-7688 and the IBM Redpaper *DB2 for z/OS: Considerations on Small and Large Packages*, REDP-4424.

Table 7-6 shows the SYSIBM.SYSPACKAGES relevant columns.

*Table 7-6 SYSPACKAGES columns of interest for distributed workload*

Column	Description
DEFERPREP	Indicates whether the PREPARE and EXECUTE are chained and sent together to the server.
ISOLATION	This option can have an impact on block fetch depending on the CURRENT DATA value
DBPROTOCOL	Whether remote access for SQL with three-part names is implemented with DRDA or DB2 private protocol access
KEEPDYNAMIC	Whether prepared dynamic statements are to be purged at each commit point. With KEEPDYNAMIC(YES), a distributed thread is not allowed to go inactive
DEFERPREPARE	Whether PREPARE processing is deferred until OPEN is executed. Y indicates the use of the bind option DEFER(PREPARE), PREPARE processing is deferred until OPEN is executed. Use DEFER(PREPARE) to improve performance for dynamic SQL so that the PREPARE and EXECUTE are chained and sent together to the server.
SQLERROR	Indicates the SQLERROR option on the most recent subcommand that bound or rebound the package. C indicates CONTINUE allowing the continuation of the package creation even in an error is found. This is useful when binding remote packages on systems where some of the referenced tables are not defined or the structure is different.

### 7.4.3 Resource Limit Facility

The Resource Limit Facility (RLF) or governor allows the system administrator to limit the amount of CPU resources either by using the reactive governor facilities of RLF or by using the predictive governing facilities.

The governor definitions are stored in the resource limit facility database DSNRLST and its tables DSNRLSTxx, where xx is any two-character alphanumeric value. If you need to know which governor table definition is currently active, use the DISPLAY RLIMIT command.

Figure 7-14 shows an example of this command executed from the OMEGAMON PE GUI interface.

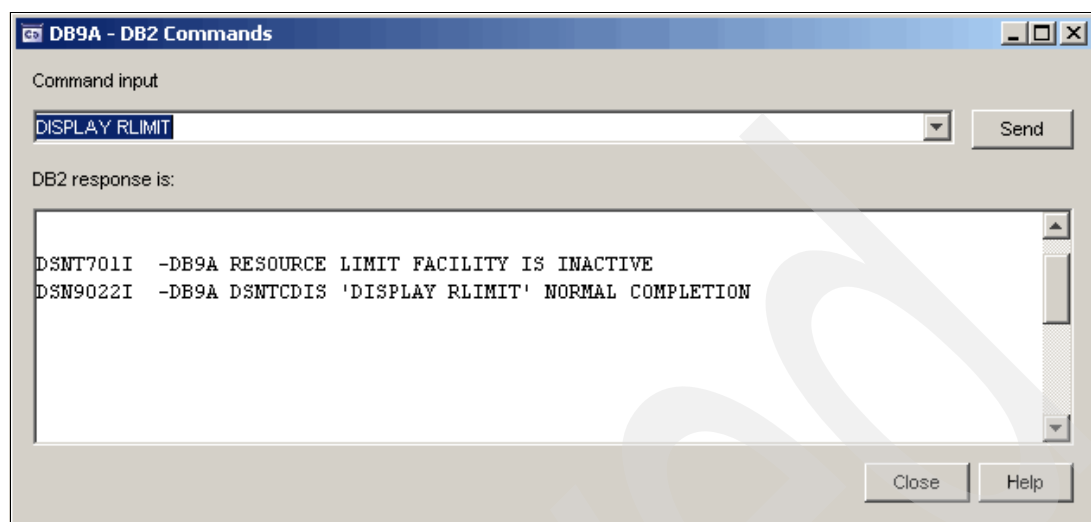


Figure 7-14 OMEGAMON PE showing the execution of a DISPLAY RLIMIT command

In this example, the RLF is not started. It can be started using the command START RLIMIT. This command accepts the optional keyword *ID=xx*. It identifies the resource limit specification table for the governor to use. *xx* is the one or two identification characters that are specified when the table is created. The full name of the table is *authid.DSNRLSTxx* or *authid.DSNRLMTxx*, where *authid* is the value that is specified in the DSNZPARM RLFAUTH.

An example of starting RLIMIT using the table SYSIBM.DSNRLST01 is shown in Figure 7-15.

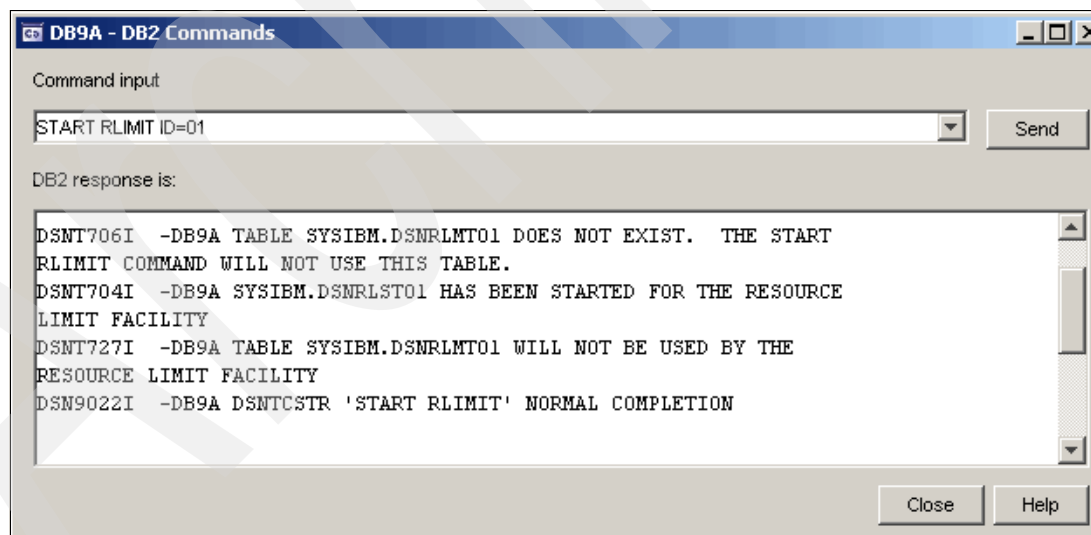


Figure 7-15 START RLIMIT example

You can stop RLF by executing the DB2 command as shown in Example 7-43.

*Example 7-43 STOP RLIMIT example*

---

```
-DB9A STOP RLIMIT
DSNT702I  -DB9A RESOURCE LIMIT FACILITY HAS BEEN STOPPED. WAS USING
SYSIBM.DSNRLST01
DSN9022I  -DB9A DSNTCSTP 'STOP RLIMIT' NORMAL COMPLETION
```

---

The DSNZPARM parameter RLF defines if the governor is to be automatically started each time DB2 is started. Parameter RLFTBL indicates the default resource limit specification table suffix (default value is 01.) Additionally, the DSNZPARM RLFERR specifies what action DB2 is to take if the governor encounters a condition that prevents it from accessing the resource limit specification table or if it cannot find a row in the table that applies to the authorization ID, the plan or package name, and the logical unit of work name of the query user. You can use this parameter for defining DB2's behavior on this case:

- ▶ Allow all dynamic SQL statements to run without limit
- ▶ Terminates all dynamic SQL statements immediately with an SQL error code
- ▶ Define a resource limit value. If this limit is exceeded, the SQL statement is terminated.

#### 7.4.4 GET\_CONFIG and GET\_SYSTEM\_INFO stored procedures

The GET\_CONFIG DB2 supplied stored procedure retrieves data server configuration information.

This data server configuration information includes the following elements:

- ▶ Data sharing group information
- ▶ DB2 subsystem status information
- ▶ DB2 subsystem parameters
- ▶ DB2 distributed access information
- ▶ Active log data set information
- ▶ The time of the last restart of DB2
- ▶ Resource limit facility information
- ▶ Connected DB2 subsystems information

**Note:** APAR PK67794 ships function enhancements for the stored procedures SYSPROC.ADMIN\_COMMAND\_DB2 and SYSPROC.GET\_CONFIG. It also ships the new stored procedure SYSPROC.ADMIN\_INFO\_SYSPARM.

Example 7-44 shows a Java program that calls this stored procedure and creates the output XML file xml\_output.xml. This program is a simplification of the example proposed in the *DB2 Version 9.1 for z/OS Administration Guide*, SC18-9840.

*Example 7-44 Example of Java program calling SYSPROC.GET\_INFO*

---

```
import java.io.*;
import java.sql.*;
public class GetConfDriver
{
    public static void main (String[] args)
    {
        Connection con = null;
        CallableStatement cstmt = null;
        String driver = "com.ibm.db2.jcc.DB2Driver";
        String url = "jdbc:db2:";
        String userid = null;
```

```

String password = null;
if (args.length != 3)
{ System.err.println("Usage: GetConfDriver <server/database><userid> <password>");
  return; }
url += args[0];
userid = args[1];
password = args[2];
try {
  Class.forName(driver);
  con = DriverManager.getConnection(url, userid, password);
  con.setAutoCommit(false);
  cstmt = con.prepareCall("CALL SYSPROC.GET_CONFIG(?,?,?,?,?,?)");
  cstmt.setInt(1, 1);
  cstmt.setInt(2, 0);
  cstmt.setString(3, "en_US");
  cstmt.setObject(4, null, Types.BLOB);
  cstmt.setObject(5, null, Types.BLOB);
  cstmt.registerOutParameter(1, Types.INTEGER);
  cstmt.registerOutParameter(2, Types.INTEGER);
  cstmt.registerOutParameter(6, Types.BLOB);
  cstmt.registerOutParameter(7, Types.BLOB);
  cstmt.execute();
  con.commit();
  SQLWarning ctstmt_warning = cstmt.getWarnings();
  if (ctstmt_warning != null) {
    System.out.println("SQL Warning: " + ctstmt_warning.getMessage());
  }
  else {System.out.println("SQL Warning: None\r\n");}
  Blob b_out = cstmt.getBlob(6);
  if(b_out != null)
  { int out_length = (int)b_out.length();
    byte[] bxml_output = new byte[out_length];
    InputStream instr_out = b_out.getBinaryStream();
    int out_len = instr_out.read(bxml_output, 0, out_length);
    FileOutputStream fxml_out = new FileOutputStream("xml_output.xml");
    fxml_out.write(bxml_output, 0, out_length );
    instr_out.close();
    fxml_out.close(); }
  Blob b_msg = cstmt.getBlob(7);
  if(b_msg != null)
  {
    int msg_length = (int)b_msg.length();
    byte[] bxml_message = new byte[msg_length];
    InputStream instr_msg = b_msg.getBinaryStream();
    int msg_len = instr_msg.read(bxml_message, 0, msg_length);
    FileOutputStream fxml_msg = new FileOutputStream(new File
("xml_message.xml"));
    fxml_msg.write(bxml_message, 0, msg_length);
    instr_msg.close();
    fxml_msg.close();
  }
} catch (SQLException sqle) {
  System.out.println("Error during CALL "
    + " SQLSTATE = " + sqle.getSQLState()
    + " SQLCODE = " + sqle.getErrorCode()
    + " : " + sqle.getMessage());}
catch (Exception e) {System.out.println("Internal Error " + e.toString());}
finally
{if(cstmt != null)
  try { cstmt.close(); } catch ( SQLException sqle)

```

```

{ sqle.printStackTrace(); }
    if(con != null)
        try { con.close(); } catch ( SQLException sqle)
{ sqle.printStackTrace(); }
    }}}

```

---

We compiled this program in a Linux on System z server having a HiperSocket connection with a z/OS partition.

Example 7-45 shows how we executed this program giving as arguments the HiperSocket IP address, the port for the z/OS server, and the user ID and password. This program connects directly to DB2 for z/OS without requiring a DB2 Connect Server, and you need to include the appropriate license file names in the Java application CLASSPATH to permit this connectivity.

---

*Example 7-45 Execution a Java program calling SYSPROC.GET\_INFO*

---

```

db2inst1@linux11:~> java GetConfDriver //10.1.1.2:12347/DB9A paolor4 123abc
SQL Warning: None

```

```

db2inst1@linux11:~>

```

---

The xml\_output.xml output file can be a challenge to read, and you may consider creating a process which cleanses the data, given the amount of valuable information that it contains. Example 7-46 shows an extract of this file.

---

*Example 7-46 Extract of xml\_output.xml file*

---

```

<key>DB2 Subsystem Specific Information</key>
-
<dict>
<key>Display Name</key>
<string>DB2 Subsystem Specific Information</string>
<key>DB9A</key>
-
<dict>
<key>Display Name</key>
<string>DB9A</string>
<key>DB2 Subsystem Status Information</key>
-
<dict>
<key>Display Name</key>
<string>DB2 Subsystem Status Information</string>
<key>DB2 Member Identifier</key>

```

---

The GET\_SYSTEM\_INFO stored procedure returns system information about the data server, including:

- ▶ Operating system information
- ▶ Product information
- ▶ DB2 MEPL
- ▶ SYSMOD APPLY status
- ▶ Workload Manager (WLM) classification rules that apply to DB2 Workload for subsystem types DB2 and DDF

Calls to this stored procedure can be implemented as described for SYSPROC.GET\_INFO.

## 7.4.5 DB2 commands

Some of the DB2 DISPLAY commands that can be used to get information about the DDF status and the distributed activity on DB2 for z/OS are described in this section.

### DISPLAY THREAD

The DB2 command DISPLAY THREAD displays current status information about DB2 threads. A DB2 thread can be an allied thread, a database access thread, or a parallel task thread. Threads can be active, inactive, indoubt, or postponed.

Distributed threads are those threads that have a connection with a remote location (active or inactive) or that had a connection with a remote location (indoubt). An allied thread and a parallel task thread can be distributed or non-distributed. A database access thread is always distributed.

The DISPLAY THREAD command allows you to select the type of information you want to display by using one or more of the following criteria:

- ▶ Active threads, inactive threads, indoubt threads, postponed threads, procedure threads, system threads, or the set of active, indoubt, postponed, and system threads are displayed according to the TYPE option entered
- ▶ Allied threads, including those threads that are associated with the address spaces whose connection names are specified
- ▶ Distributed threads, including those threads that are associated with a specific remote location
- ▶ Detailed information about connections with remote locations
- ▶ A specific logical unit of work ID (LUWID)

The information that is returned by the DISPLAY THREAD command reflects a dynamic status. When the information is displayed, it is possible that the status has changed. Moreover, the information is consistent only within one address space and is not necessarily consistent across all address spaces displayed.

The syntax and options of this command are described in details in *DB2 Version 9.1 for z/OS Command Reference*, SC18-9844.

Example 7-47 shows an example of the DISPLAY THREAD command. This particular case shows details of a thread using trusted contexts.

*Example 7-47 DISPLAY THREAD output example*

---

```
DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -DB9A ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN      ASID  TOKEN
TSO       T  *    3 PAOLOR4      PAOLOR4          00C2   105
TSO       N          7 RAJESH      RAJESH          00C4    0
V485-TRUSTED CONTEXT=RAJESH,
        SYSTEM AUTHID=RAJESH,
        ROLE=*
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DB9A DSNVDT '-DIS THD' NORMAL COMPLETION
```

---

The TYPE(INACTIVE) option is interesting when looking for distributed workload information. This option displays only inactive database access threads and connections that are connected by a network connection to another system. An inactive thread is idle and waits for

a new unit of work to begin from that system. Use qualifiers such as complete location names or LUWIDs with this option. When there are large numbers of inactive database access threads, unqualified display requests can temporarily change the DB2 working set, which can temporarily affect the performance of active threads

Example 7-48 shows a partial output for the command `DIS THD(*) TYPE(INACTIVE)`.

*Example 7-48 Display of INACTIVE THREADS*

---

```

DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV424I  -DB9A INACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN      ASID TOKEN
...
SERVER    R2      0 db2bp          PAOLOR4  DISTSERV  00A7  3219
V437-WORKSTATION=linux11, USERID=paolor4,
        APPLICATION NAME=db2bp
V442-CRTKN=9.12.4.121.47323.090427005153
V445-G90C0479.B8DB.C418F0959823=3219 ACCESSING DATA FOR
        ::9.12.4.121
DISPLAY INACTIVE REPORT COMPLETE
DSN9022I  -DB9A DSNVDT '-DIS THREAD' NORMAL COMPLETION

```

---

The `LOCATION` option can be used for display only distributed threads of the specified type that either have (active or inactive threads) or had (indoubt threads) a remote connection with the specified location name.

DB2 does not receive a location name from requesters that are not DB2 for z/OS subsystems. For instance, to display all threads of the remote location shows in Example 7-48, you use the `-DIS THD(*) LOCATION(::9.12.4.121)` command. Example 7-49 shows a partial output of this command.

*Example 7-49 DIS THD(\*) LOCATION(::9.12.4.121) command output*

---

```

DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -DB9A ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN      ASID TOKEN
SERVER    RA *    27 db2bp          PAOLOR4  DISTSERV  00A7  7292
V437-WORKSTATION=linux11, USERID=paolor4,
        APPLICATION NAME=db2bp
V442-CRTKN=9.12.4.121.41559.090428150147
V445-G90C0479.A257.C41AF06AE488=7292 ACCESSING DATA FOR
        ::9.12.4.121
SERVER    RA *    27 db2bp          PAOLOR4  DISTSERV  00A7  7291
V437-WORKSTATION=linux11, USERID=paolor4,
        APPLICATION NAME=db2bp
V442-CRTKN=9.12.4.121.41558.090428150147
V445-G90C0479.A256.C41AF06ADAF8=7291 ACCESSING DATA FOR
        ::9.12.4.121
....
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DB9A DSNVDT '-DIS THD' NORMAL COMPLETION

```

---

Additional variants and application examples of this command and its options can be found in the IBM documentation.



## DISPLAY LOCATION

The DB2 command DISPLAY LOCATION displays various information about the specified remote locations. If you specify the DETAIL option, each line can be followed by information regarding conversations owned by DB2 system threads that are communicating with the location. The information returned by the DISPLAY LOCATION command reflects a dynamic status. By the time the information is displayed, it is possible that the status has changed.

Example 7-50 shows the output of this command.

*Example 7-50 DISPLAY LOCATION example*

---

```
DSNL200I  -DB9A DISPLAY LOCATION REPORT FOLLOWS-
LOCATION                                PRDID    REQSTR  SERVER  CONVS
::9.12.4.121                          0        0      25     25
::9.12.5.149                          0        0       9      9
::9.12.6.70                           0        0       2      2
DB9C                                   DSN09015  0        0       0
  ::FFFF:9.12.4.104
DB9C                                   DSN09015  0        0       0
  ::9.12.6.9
DISPLAY LOCATION REPORT COMPLETE
***
```

---

You can exploit the following information:

► **REQSTR**

The number of requester connections from the local subsystem that are accessing the remote system. In this example, the local system is not accessing any remote system.

► **SERVER**

The number of connections from the remote system that are accessing the local system as a server.

► **CONVS**

The total number of conversations related to the partner system. This value may include DB2 system related connections that are not reflected in the REQSTR and SERVER values.

## DISPLAY DDF

The DISPLAY DDF command displays information regarding the status and configuration of DDF, as well as statistical information regarding connections or threads controlled by DDF.

The syntax of this command only accepts the DETAIL option, as shown in Example 7-51.

*Example 7-51 DIS DDF syntax*

---

```
>>-DISPLAY DDF--+-+-----+-----><
                  '-DETAIL-'
```

---

Example 7-52 on page 318 shows an example of DIS DDF for a stand-alone DB2 server after applying APAR PK80474. The RESYNC domain is no longer displayed because it is identical to the SQL domain. In a data sharing configuration, the DIS DDF command has been enhanced to display the server list, Group IP address, member specific IP address and defined location aliases when PK80474 is applied. More details are provided in Example 6-2 on page 236.

#### Example 7-52 DIS DDF command example

```
DSNL080I -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I TCPPORT=12347 SECPORT=12349 RESPORT=12348 IPNAME=-NONE
DSNL085I IPADDR=:9.12.6.70
DSNL086I SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Some of the relevant information in the output is:

- ▶ **STATUS:** The operational status of DDF.
- ▶ **LOCATION:** The location name of DDF.
- ▶ **LUNAME:** The fully qualified LUNAME of DDF.
- ▶ **GENERICLU:** The fully qualified generic LUNAME of DDF.
- ▶ **IPADDR:** The IP address of DDF.
- ▶ **TCPPORT:** The SQL listener port used by DDF. Use this port when configuring a connection to this DB2 subsystem.
- ▶ **SECPORT:** The TCP/IP SQL listener port used by DDF to accept inbound SSL connections. This information is not shown in this example as this DB2 subsystem is not currently configured for accepting SSL connections.
- ▶ **RESPORT:** The TCP/IP port number used by this DB2 subsystem to accept incoming two-phase commit resynchronization requests.

Additionally, the message DSNL087I displays column headers for a table that contains the alias names and their associated ports. Message DSNL088I displays each alias name, and its associated port (if any), defined in the BSDS DDF record. The DSNL087I and DSNL088I messages are provided only if alias information is present in the BSDS DDF record. These two messages are not shown in this example.

Example 7-53 shows the output of this command when using the DETAIL option. This option displays additional statistics and configuration information.

#### Example 7-53 DIS DDF DETAIL example

```
DSNL080I -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I TCPPORT=12347 SECPORT=12349 RESPORT=12348 IPNAME=-NONE
DSNL085I IPADDR=:9.12.6.70
DSNL086I SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL086I RESYNC  DOMAIN=wtsc63.itso.ibm.com
DSNL090I DT=I  CONDBAT=    300 MDBAT=    100
DSNL092I ADBAT=    100 QUEDBAT=    349 INADBAT=      0 CONQUED=    24
DSNL093I DSCDBAT=      0 INACONN=    27
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Notice the following relevant configuration information about this output:

- ▶ **DT=I** specifies the current CMSTAT setting, in this case I = INACTIVE
- ▶ **CONDBAT=300**
- ▶ **MDBAT=100**

The following information describes the current DDF workload status:

► ADBAT=100

► QUEDBAT=349

This value reflects a cumulative counter that is always incremented when the MDBAT limit has been reached. A value and a non-zero value suggests that performance and throughput may have been affected, in which case it might be appropriate to consider increasing the MAXDBAT DSNZPARM parameter value.

► DSCDBAT=0

This is the current number of disconnected database access threads. This value only applies if DDF INACTIVE support is enabled (DT=I). This value is effectively the number of DBAT pool threads. DBAT pool threads are the database access threads that are available to service queued connections requests.

► INACONN=27

This is the number of inactive connections and only applies if DDF supports INACTIVE connections. This value represents connections with clients where the client last ended a unit of work causing DB2 to change the state of the connection to inactive. That means that the DBAT is separated from the connection, and the connection becomes inactive while the DBAT is made available to service new client connections or to service new requests on existing client connections. Any connections reflected here can also be observed in the DISPLAY THREAD TYPE(INACTIVE) command report.

► CONQUED=24

This is the current number of connection requests that have been queued and are waiting to be serviced. This value only applies if the dt value that is specified in the DSNL090I message indicates that DDF INACTIVE support is enabled. These connections can represent newly attached connections, or inactive connections for which a new request has arrived from the client. These requests cannot be processed when one of the following conditions occur:

- The maximum number of database access threads has been reached (adbat value is equal to or greater than the DSNL090I mdbat value) and there are no disconnected database access threads to process the request (see DSNL093I dscdbat value). Processing continues for these *conquered connections* when another active database access thread releases its slot by either terminating or by moving to the inactive state (see the inadbat value or the DSNL093I inaconn value).
- The DDF is suspended or suspending (see DSNL081I status value). Processing continues when the DDF is started (resumed).

Example 7-54 shows the output of the command DIS THD(\*) TYPE(INACTIVE).

*Example 7-54 DIS THD(\*) TYPE(INACTIVE) command output*

---

```

DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV424I  -DB9A INACTIVE THREADS -
NAME      ST A  REQ ID          AUTHID  PLAN    ASID TOKEN
SERVER    R2      0 db2bp          PAOL0R4  DISTSERV 00A7  233
V437-WORKSTATION=kodiak.itso.ibm., USERID=paolor4,
APPLICATION NAME=db2bp
V442-CRTKN=9.12.5.149.52594.090425220432
V445-G90C0595.CD72.C41789289596=233 ACCESSING DATA FOR
::9.12.5.149
SERVER    R2      0 db2jcc_appli RAJESH  DISTSERV 00A7  2616
V437-WORKSTATION=wtsc63.itso.ibm., USERID=RAJESH,
APPLICATION NAME=db2jcc_application
V445-G90C0646.G5AA.C418D8B327A2=2616 ACCESSING DATA FOR

```

```

::9.12.6.70
SERVER R2 0 db2bp PAOL0R4 DISTSERV 00A7 2835
V437-WORKSTATION=linux11, USERID=pao1or4,
APPLICATION NAME=db2bp
V442-CRTKN=9.12.4.121.58276.090427001247
V445-G90C0479.E3A4.C418E7D81095=2835 ACCESSING DATA FOR
::9.12.4.121

```

---

The DISPLAY DDF DETAIL command is especially useful because it reflects the presence of new inbound connections that are not reflected by other commands. For example, if DDF is in INACTIVE MODE, as denoted by DT=I in Example 7-53 on page 318, and DDF is stopped with mode SUSPEND, or the maximum number of active database access threads has been reached, new inbound connections are not yet reflected in the DISPLAY THREAD report.

However, the presence of these new connections is reflected in the DISPLAY DDF DETAIL report. Specific details regarding the origin of the connections, such as the client system IP address or LU name, are not available until the connections are actually associated with a database access thread.

The command STOP DDF MODE(SUSPEND) suspends all DDF threads in the following manner:

- ▶ Keeping inactive DDF threads inactive until a subsequent START DDF command is issued
- ▶ Terminating all DDF pool threads
- ▶ Preventing inbound DDF work from starting

This mode is intended to be used when locking conflicts exist between CREATE, ALTER, DROP, GRANT, or REVOKE operations and client access to data. Requests that normally cause work to be dispatched (including requests for new connections) are queued. Outbound DDF processing is not affected by this command.

The syntax of the STOP DDF command is shown in Example 7-55.

*Example 7-55 STOP DDF command syntax*

```

.-QUIESCE-----
>>-STOP DDF--MODE(-+-----+--)-><
      +-FORCE-----+
      '-SUSPEND-+-CANCEL(n)-+-'
              '-WAIT(n)---'

```

---

Example 7-56 shows the system log messages indicating that DDF was stopped in suspend mode successfully.

*Example 7-56 STOP DDF output*

```

DSNL069I -DB9A DSNLSSRS DDF IS SUSPENDING
DSNL066I -DB9A DSNLSSRS STOP DDF MODE(SUSPEND) COMPLETE

```

---

Example 7-57 on page 321 shows how the command DIS DDF indicates that the DDF activity is suspended using the STATUS=SUSPND field. This command was executed while remote activity was being executed on the system, and you can notice that after the executions of this command, the number of active DBATS=0, indicated by the field ADBAT=0.

*Example 7-57 DISPLAY DDF output (suspending)*

---

```
DSNL080I  -DB9A DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=SUSPND
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DB9A              USIBMSC.SCPDB9A  -NONE
DSNL084I  TCPPOPT=12347 SECPORT=12349 RESPORT=12348 IPNAME=-NONE
DSNL085I  IPADDR=:9.12.6.70
DSNL086I  SQL      DOMAIN=wtsc63.itso.ibm.com
DSNL090I  DT=I  CONDBAT=    300 MDBAT=   100
DSNL092I  ADBAT=    0 QUEDBAT=   667 INADBAT=    0 CONQUED=    9
DSNL093I  DSCDBAT=    0 INACONN=    5
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

---

Example 7-58 is the output of the DIS LOCATION command executed when DDF was in suspended status. As indicated by the columns REQSTR, the local DB2 was acting as a requester against a remote DB2 for z/OS. Outbound DDF processing is not affected when DDF is in status STATUS=SUSPND.

*Example 7-58 DISPLAY LOCATION output*

---

LOCATION	PRDID	REQSTR	SERVER	CONVS
::9.12.4.121		0	25	25
::9.12.6.70		0	2	2
DB9C	DSN09015	1	0	1
::FFFF:9.12.4.104				
DB9C	DSN09015	0	0	0
::9.12.6.9				

DISPLAY LOCATION REPORT COMPLETE

---

Example 7-59 shows the output of the START DDF command executed when DDF was suspended. The message DSNL068I indicates that DDF has resumed normal processing.

*Example 7-59 DISPLAY DDF after RESUME*

---

```
DSNL070I  -DB9A DSNLSSRS DDF IS RESUMING
DSNL068I  -DB9A DSNLSSRS START DDF (RESUME PROCESSING) COMPLETE
```

---

Archived

## Problem determination

In a day-to-day application development environment, programs are routinely developed, tested, and migrated to production. Most shops have internal procedures to stress-test the application before every major release. However, in spite of the stringent quality checking, it is not uncommon in production environments to encounter problems related to performance or connectivity to the database server.

Problem determination in an n-tier infrastructure can be challenging, as several heterogeneous components must be analyzed. Typically, the know-how and expertise required do not reside on a single team or person, and an inter-platform interaction is required.

In this chapter, we discuss the options available for problem determination in a n-tier environment from a system, rather than an application, point of view. We summarize the tools and utilities available on the different components of a n-tier architecture and provide guidelines about how to start a problem determination exercise and which tools to exploit for specific scenarios.

This chapter contains the following sections:

- ▶ “Traces at DB2 Client and DB2 Gateway” on page 324
- ▶ “Accounting for distributed data with the EXCSQLSET command” on page 343
- ▶ “Network analyzer” on page 355
- ▶ “DB2 for z/OS tracing capabilities” on page 363

## 8.1 Traces at DB2 Client and DB2 Gateway

In this section we look at:

- ▶ The n-tier message communication
- ▶ CLI traces
- ▶ DRDA traces
- ▶ JDBC traces

### 8.1.1 The n-tier message communication

Figure 8-1 shows a simplified implementation of an n-tier architecture. In this figure, DB2 clients (either thin or fat clients, or Web or application servers) connect through a network to a communication server. In some configurations, this communication server is a DB2 Connect Server.

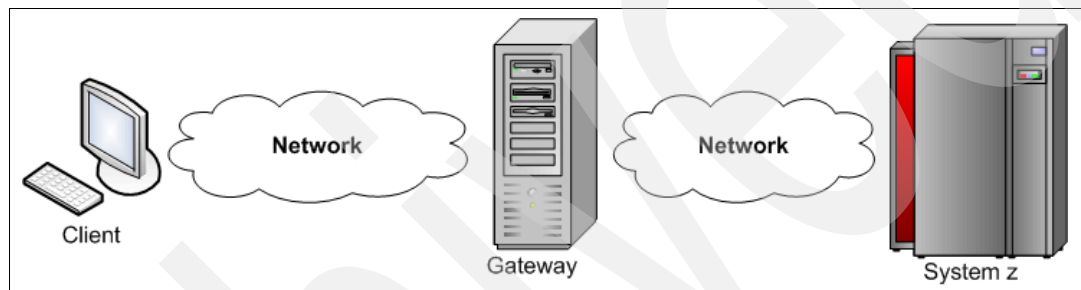


Figure 8-1 3-tier architecture simplified

For some other configurations, no intermediate gateway is required, the IBM Data Server Drivers or Clients allow applications to talk DRDA directly to DB2 for z/OS with equivalent or better functions. This is represented in Figure 8-2 and constitutes a 2 tier architecture.

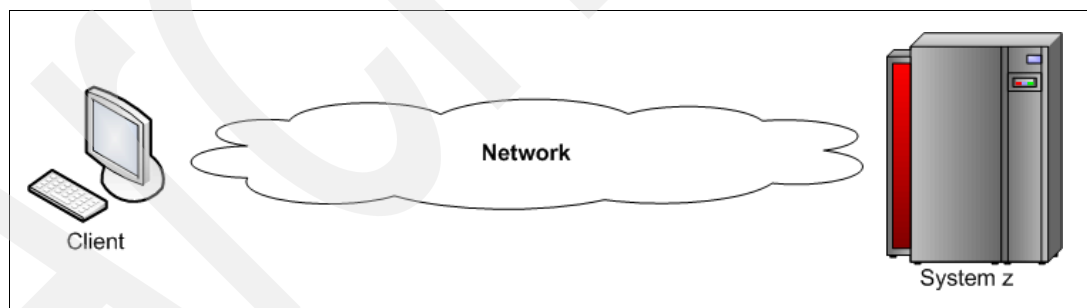


Figure 8-2 2-tier architecture simplified

Refer to Chapter 2, “Distributed database configurations” on page 33 for more information about configuration options.

In the context of an n-tier architecture, performance problems can happen at any tier. The communication can have increased network related latency between the client and DB2 Connect Server or between the DB2 Connect Server and the server. This can happen because the routing policy was changed by the network team or a change in the application code caused the elapsed time spent at the server side to increase substantially. Changes in how the programs are compiled on the DB2 Connect Server server can cause extra messages to be sent, increasing the network traffic and causing a performance issue.



The purpose of the following sections is to describe the available traces on each one of the components of a n-tier architecture. Figure 8-3 represents an overview of the available traces and tools by component.

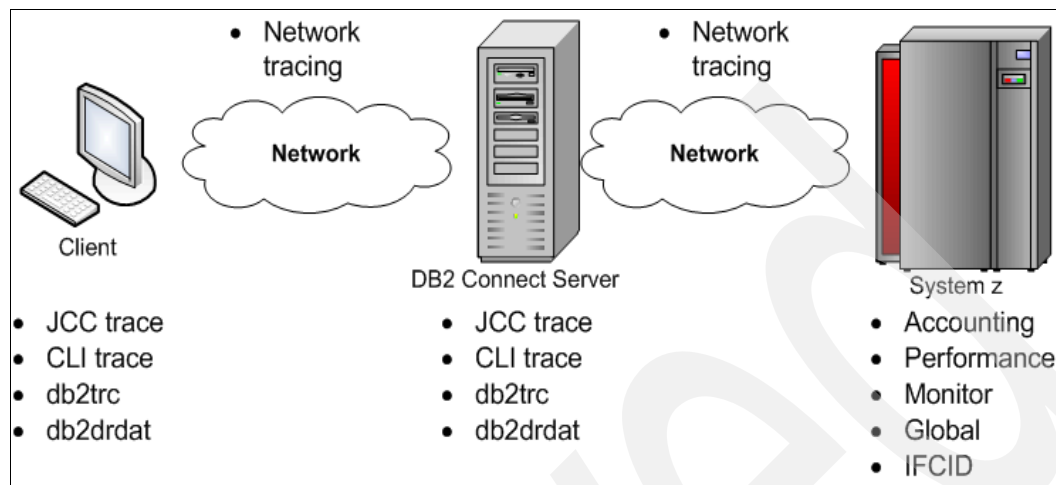


Figure 8-3 Traces and tools in a n-tier environment

Table 8-1 summarizes the traces that are available in a distributed component of an n-tier architecture. The table shows the traces available by Client/Driver type.

Table 8-1 Traces available on distributed components

Client/Driver	Trace available	Description of what the trace contains
IBM Data Server Driver for JDBC and SQLJ (Type 4)	JCC Trace	It contains both JCC driver trace and DRDA trace. JCC trace contains both JCC driver trace and DRDA trace only when TRACE_ALL is specified.
IBM Data Server Driver for ODBC, CLI and .NET	CLI Trace, db2trc	CLI trace contains the driver trace. db2trc contains db2 client side buffers and DRDA buffers. Note that db2drdat is available with data server drivers shipped with 9.5 FixPack 4 and higher.
All other Data Server Clients, DB2 Connect, DB2 Enterprise Server Edition (ESE), and so forth	CLI trace, db2trc, db2drdat	CLI trace + db2trc + db2drdat. db2drdat contains only DRDA buffers.

## 8.1.2 CLI traces

The CLI trace captures information about applications that access the DB2 CLI driver. This trace offers little information about the internal workings of the DB2 CLI driver. This type of trace is applicable for situations where a problem is encountered in the following areas:

- ▶ A CLI application
- ▶ An ODBC application (ODBC applications use the DB2 CLI interface to access DB2)
- ▶ DB2 CLI stored procedures
- ▶ JDBC applications and stored procedures

When diagnosing ODBC applications, it is often easier to determine the problem by using an ODBC trace or DB2 CLI trace. If you are using an ODBC driver manager, it will likely provide the capability to take an ODBC trace. Consult your driver manager documentation to

determine how to enable ODBC tracing. DB2 CLI traces are DB2-specific and often contain more information than a generic ODBC trace. CLI and ODBC traces are usually quite similar, listing entry and exit points for all CLI calls from an application, and including any parameters and return codes to those calls.

The DB2 JDBC Type 2 driver for LUW (DB2 JDBC Type 2 driver) depends on the DB2 CLI driver to access the database. Consequently, Java developers might also want to enable DB2 CLI tracing for additional information about how their applications interact with the database through the various software layers. DB2 JDBC and DB2 CLI trace options (though both set in the db2cli.ini file) are independent of each other. This driver, also called the CLI existing driver, has been deprecated.

The db2cli.ini file is the DB2 call level interface (CLI) configuration file. It contains various keywords and values that can be used to modify the behavior of the DB2 CLI and the applications using it. The file is divided into sections, with each section corresponding to a database alias name. The COMMON section affects all the database aliases.

IBM Data Server Drivers can also use the db2dsdriver configuration file. The configuration file db2dsdriver.cfg contains database directory information and client configuration parameters in a human-readable format. Refer to 5.4.3, “db2cli.ini and db2dsdriver.cfg” on page 206 for details.

**Note:** A CLI trace affects ALL the transactions that are executed through the impacted driver or server unless you specify the process to be traced. This can be achieved by using the keyword TracePIDList. This parameter restricts the process IDs for which the CLI/ODBC trace will be enabled.

To activate a CLI trace, the following five configuration keywords need to be added to the COMMON section of the CLI configuration:

► Trace=1.

This setting turns on the DB2 CLI/ODBC trace facility. When you set this configuration keyword to 1, CLI/ODBC trace records are appended to the file indicated by the TraceFileName configuration parameter or to files in the subdirectory indicated by the TracePathName configuration parameter. Trace has no effect if you do not set either TraceFileName or TracePathName.

► TracePathName=<path>.

This setting specifies the subdirectory to be used to store individual DB2 CLI/ODBC trace files. Each thread or process that uses the same DLL or shared library will have a separate DB2 CLI/ODBC trace file created in the specified directory. A concatenation of the application process ID and the thread sequence number is automatically used to name trace files. No trace will occur, and no error message will be returned, if the subdirectory given is invalid or if it cannot be written to. This option is only used when the Trace option is turned on.

► TraceComm=1.

This setting specifies whether information about each network request is included in the trace file. When TraceComm is set on then the following information about each network request will be included in the trace file:

- Which DB2 CLI functions are processed completely on the client and which DB2 CLI functions involve communication with the server.
- The number of bytes sent and received in each communication with the server.
- The time spent communicating data between the client and server.

► TraceFlush=1.

This setting forces a write to disk after n CLI/ODBC trace entries. By default, TraceFlush is set to 0 and each DB2 CLI trace file is kept open until the traced application or thread terminates normally. If the application terminates abnormally, some trace information that was not written to the trace log file may be lost. Set this keyword to a positive integer to force the DB2 CLI driver to close and re-open the appropriate trace file after the specified number of trace entries. For this example, 1.

► TraceTimeStamp=1.

This setting specifies what type of time stamp information, if any, is recorded in the CLI/ODBC trace. The following settings indicate what type of time stamp information is captured in the trace file:

- 0: no time stamp information
- 1: processor ticks<sup>1</sup> and ISO time stamp (absolute execution time in seconds and milliseconds, followed by a time stamp)
- 2: processor ticks (absolute execution time in seconds and milliseconds)
- 3: ISO time stamp

You can find the definition of the CLI/ODBC configuration keywords listed by category at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.apdv.cli.doc/doc/r0007964.html>

In the following sections we look at the following topics:

- Starting CLI traces
- Working with CLI traces

## Starting CLI traces

A CLI trace is enabled by adding specific entries to the db2cli.ini file.

It is important to consider that there are many keywords that can be added to the db2cli.ini file that can affect application behavior. These keywords can resolve or be the cause of application problems. There are also some keywords that are not covered in the CLI documentation. Those are only available from DB2 Support. If you have keywords in your db2cli.ini file that are not documented, it is likely that they were recommended by the DB2 support team.

Before you start the trace, as a preliminary task, you need to create or select a path for the trace files. It is important to create or use a path that every user can write to. For example, if you create a directory for this purpose on UNIX with the command `mkdir /tmp/clitrace`, you could apply the `chmod 777 /tmp/clitrace` command for getting adequate permissions.

In this section we examine the following options to start a CLI trace:

- Enabling CLI traces using the DB2 Configuration Assistant
- Enabling CLI traces using CLP commands
- Enabling CLI traces editing the db2cli.ini file

### ***Enabling CLI traces using the DB2 Configuration Assistant***

Figure 8-4 shows how you can select the option of changing a cataloged database's CLI Settings from the DB2 Configuration Assistant. For this example, we modified the CLI settings of a cataloged DB2 for z/OS database.

<sup>1</sup> Processor ticks (time) is different from actual wall clock time because it does not include any time spent waiting for I/O or when some other process is running.

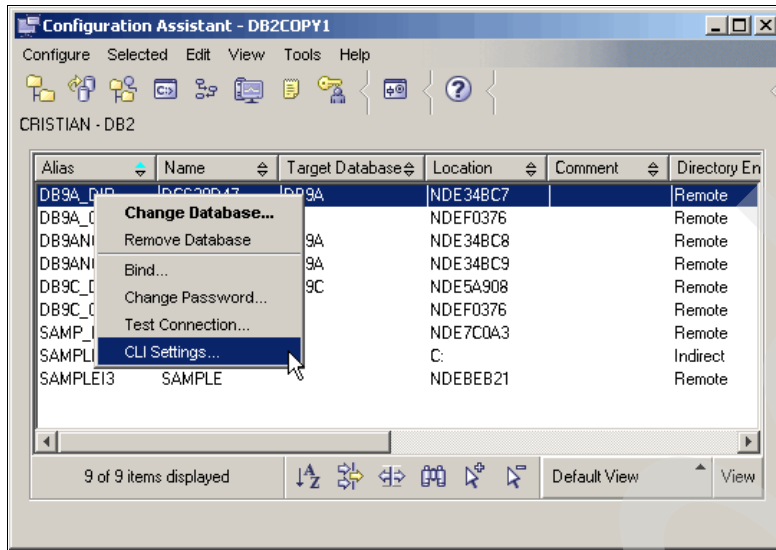


Figure 8-4 CLI settings in the Configuration Assistant

Navigate to **CLI Settings** → **Settings** → **Add** to get to the “Add CLI Parameter” panel (Figure 8-5). In this panel you can enable the trace and add the five trace parameters described above. This panel also provides a description of the parameters and some more information.

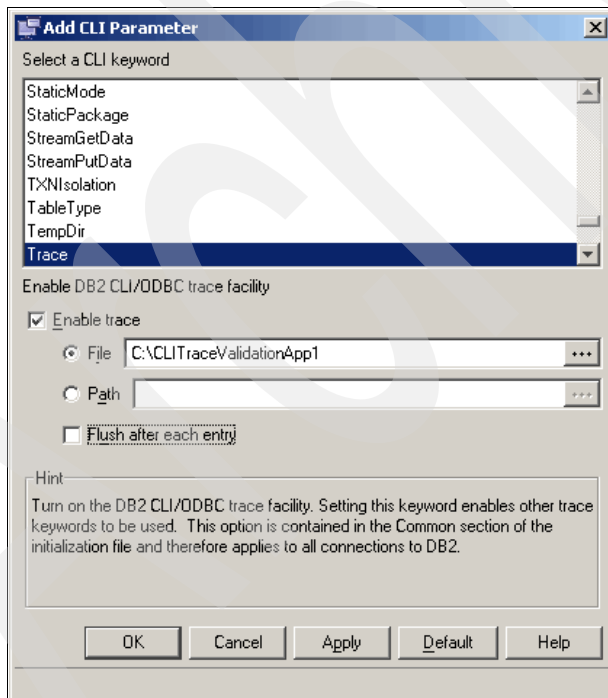


Figure 8-5 Add CLI parameters in the Configuration Assistant

After entering the required parameters, you can browse them as shown in Figure 8-6. This panel shows the current CLI settings. You may decide to keep the definitions after creating traces for a problem determination and deactivating the traces by changing the parameter Trace to 0. You are not required to remove the other trace related definitions, but it makes it easier to activate the traces in the future.

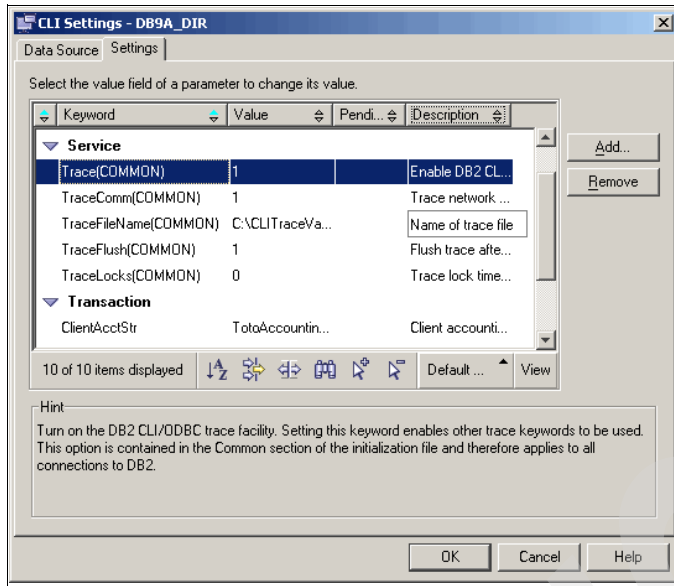


Figure 8-6 CLI Settings panel

Windows user can also change the CLI settings by using the CLI/ODBC Setting panels from the Windows Control panel. Navigate to **Start → Settings → Control Panel → Administrative Tools → Data Sources (ODBC)**.

An example of the panel is shown in Figure 8-7.

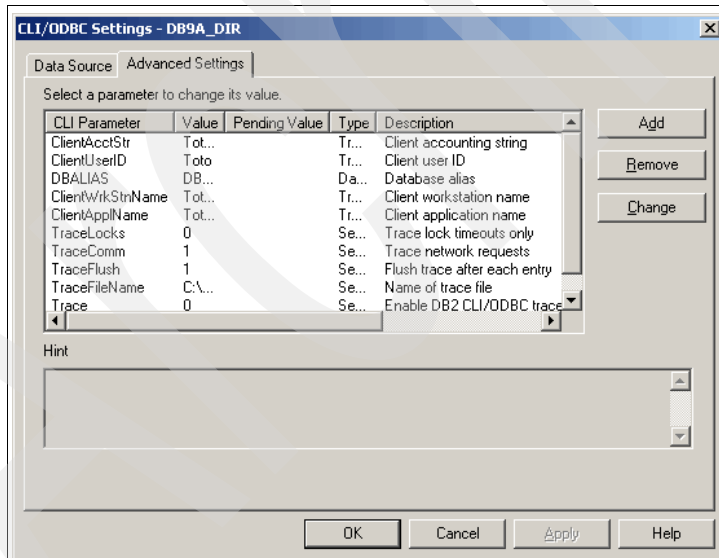


Figure 8-7 CLI/ODBC Setting, Windows Control panel

### Enabling CLI traces using CLP commands

You can update the db2cli.ini file by using UPDATE CLI CFG and query the current CLI settings by using the GET CLI CFG commands. This method presents the following advantages:

- ▶ It protect you from syntax errors that can occur if editing the db2cli.ini file directly.
- ▶ It can be used instead of GUIs not always available.

- It saves you from finding where the actually used db2cli.ini file is located.
- It gives better control of changes on servers containing multiple instances by using the instance owner user for introducing changes.

The GET CLI CONFIGURATION command lists the contents of the db2cli.ini file. This command can list the entire file or a specified section. Example 8-1 shows its syntax.

*Example 8-1 GET CLI CONFIGURATION syntax*

---

```
>>-GET CLI--+-CONFIGURATION-+--+-----+----->
      +-CONFIG-----+ '-AT GLOBAL LEVEL-'
      '-CFG-----'
```

---

```
>--+-----+-----><
      '-FOR SECTION--section-name-'
```

---

Example 8-2 shows the output of its execution. This GET CLI CONFIGURATION command is executed before executing the commands that are required for starting the CLI trace.

*Example 8-2 GET CLI CONFIGURATION output example*

---

```
$ db2 get cli configuration
```

```
Section: tstcli1x
```

```
-----
uid=userid
pwd=*****
autocommit=0
TableType='TABLE','VIEW','SYSTEM TABLE'
```

```
Section: tstcli2x
```

```
-----
SchemaList='OWNER1','OWNER2',CURRENT SQLID
```

```
Section: MyVeryLongDBALIASName
```

```
-----
dbalias=dbalias3
SysSchema=MYSHEMA
```

---

The UPDATE CLI CONFIGURATION command updates the contents of a specified section of the db2cli.ini file. You can execute this sequence of commands to enable CLI tracing:

1. db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
2. db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName <path>
3. db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
4. db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
5. db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3

Example 8-3 on page 330 shows an example of executing these commands on an UNIX server.

*Example 8-3 Execution of UPDATE CLI CFG commands*

---

```
$ db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
DB20000I The UPDATE CLI CONFIGURATION command completed successfully.
$ db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
DB20000I The UPDATE CLI CONFIGURATION command completed successfully.
$ db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName /tmp/clitrace/
```

```
DB20000I The UPDATE CLI CONFIGURATION command completed successfully.
$ db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
DB20000I The UPDATE CLI CONFIGURATION command completed successfully.
$ db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3
DB20000I The UPDATE CLI CONFIGURATION command completed successfully.
```

---

Example 8-4 shows the execution of the GET CLI CFG command after the application of the UPDATE CLI CFG commands. Note the creation of the new section *COMMON* which contains the entered parameters.

*Example 8-4 GET CLI CFG command output after changes*

---

```
$ db2 get cli configuration

Section: tstcli1x
-----
uid=userid
pwd=*****
autocommit=0
TableType='TABLE','VIEW','SYSTEM TABLE'

Section: tstcli2x
-----
SchemaList='OWNER1','OWNER2',CURRENT SQLID

Section: MyVeryLongDBALIASName
-----
dbalias=dbalias3
SysSchema=MYSHEMA

Section: COMMON
-----
TraceTimeStamp=3
TraceFlush=1
TracePathName=/tmp/clitrace/
TraceComm=1
Trace=1
```

---

### **Enabling CLI traces editing the db2cli.ini file**

By default, the location of the DB2 CLI/ODBC configuration keyword file is the sqllib directory of Windows operating systems, and the sqllib/cfg directory of the database instance running the CLI/ODBC applications on Linux and UNIX operating systems. Example 8-5 on page 331 shows the commands that can be used to find the location of the db2cli.ini in an AIX server. In this example, several products were installed and each instance owns its own db2cli.ini file.

*Example 8-5 How to find the db2cli.ini file in an AIX server*

---

```
# whoami
root
# cd /
# find . -name db2cli.ini
./home/db2inst1/sqllib/cfg/db2cli.ini
./home/db2inst2/sqllib/cfg/db2cli.ini
./home/db2inst3/sqllib/cfg/db2cli.ini
./opt/IBM/db2/V9.5/cfg/db2cli.ini
./opt/IBM/db2/V9.5_01/cfg/db2cli.ini
```

```
./opt/IBM/db2/V9.5_ESE/cfg/db2cli.ini
```

---

The environment variable DB2CLIINIPATH can also be used to override the default and specify a different location for the file. Example 8-6 shows the usage of the db2 command **db2set** to display the value of the environment variable DB2CLIINIPATH. You need to be logged as the instance owner or have sourced the /sqlib/db2profile file in your profile to execute this command. This example shows that this environment variable was not set, so there is no override of the db2cli.ini default file location.

*Example 8-6 Using the db2set command to display the variable DB2CLIINIPATH*

---

```
# su - db2inst1
$ db2set DB2CLIINIPATH
DBI1303W Variable not set.
Explanation:
The variable was not set in the profile registry.
User response:
No further action is required.
$
```

---

To enable the CLI trace, you need to edit the db2cli.ini file that belongs to the instance to be analyzed by adding the section shown in Example 8-7.

*Example 8-7 COMMON section of db2cli.ini*

---

```
[COMMON]
Trace=1
TracePathName=<path>
TraceComm=1
TraceFlush=1
TraceTimeStamp=1
```

---

## Working with CLI traces

You can find extensive information about how to use a CLI trace in the IBM DB2 for Linux, UNIX, and Windows Information Center at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>

The following topics are covered in that documentation:

- ▶ Interpreting input and output parameters in CLI trace files
- ▶ Analyzing Dynamic SQL in CLI traces
- ▶ Interpreting timing information in CLI traces
- ▶ Interpreting unknown values in CLI traces
- ▶ Interpreting multi-threaded CLI trace output

Example 8-8 is an extract of a CLI trace collected in a Windows client when accessing a DB2 9 for z/OS.

*Example 8-8 CLI trace example*

---

```
[ Process: 5640, Thread: 3044 ]
[ Date & Time:      04/28/2009 22:39:58.465029 ]
[ Product:         QDB2/NT DB2 v9.5.0.808 ]
[ Level Identifier: 03010107 ]
[ CLI Driver Version: 09.02.0000 ]
[ Informational Tokens: "DB2 v9.5.0.808","s071001","NT3295","Fixpack 0" ]
[ Install Path:    C:\PROGRA~1\IBM\SQLLIB ]
```



```
[ db2cli.ini Location: C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY1\db2cli.ini ]
[ CLI Driver Type:      IBM DB2 Application Runtime Client ]

[1240972988.673504 - 04/28/2009 22:43:08.673504] SQLPrepareW( hStmt=1:1, pszSqlStr="select name from
sysibm.systables", cbSqlStr=33 )
[1240972988.688430 - 04/28/2009 22:43:08.688430]      ---> Time elapsed - +4.367000E-003 seconds
[1240972988.716118 - 04/28/2009 22:43:08.716118] ( StmtOut="select name from sysibm.systables FOR
FETCH ONLY" )
[1240972988.724024 - 04/28/2009 22:43:08.724024]

[1240972988.731988 - 04/28/2009 22:43:08.731988] SQLPrepareW( )
[1240972988.736115 - 04/28/2009 22:43:08.736115]      <--- SQL_SUCCESS   Time elapsed - +6.261100E-002
seconds

[1240972988.739733 - 04/28/2009 22:43:08.739733] SQLNumParams( hStmt=1:1, pcPar=&0c57fd68 )
[1240972988.750079 - 04/28/2009 22:43:08.750079]      ---> Time elapsed - +3.618000E-003 seconds

[1240972988.753422 - 04/28/2009 22:43:08.753422] SQLNumParams( pcPar=0 )
[1240972988.761842 - 04/28/2009 22:43:08.761842]      <--- SQL_SUCCESS   Time elapsed - +2.210900E-002
seconds
...

```

---

### 8.1.3 DRDA traces

The following topics are discussed in this section:

- ▶ db2drdat – DRDA trace command
- ▶ DRDA trace output
- ▶ DRDA trace output file analysis

#### db2drdat – DRDA trace command

DB2 talks DRDA, refer to Chapter 1, “Architecture of DB2 distributed systems” on page 3 and the bibliography for details about the DRDA architecture and components.

The **db2drdat** command allows you to capture the DRDA data stream exchanged between a DRDA Application Requestor (AR) and the DRDA Application Server (AS). This tool is most often used for problem determination, but it also provides a lot of useful performance information. DRDA traces can be executed in a Client (including a DB2 Connect Server), DRDA traffic can be followed flowing in the network, and DRDA traces can be started in z/OS (IFCID 180 and 184). You can see DRDA traces as a cross platform monitoring tool for DRDA related problem determination.

If a receive buffer contains SQLCA information, it will be followed by a formatted interpretation of this data and labeled SQLCA. The SQLCODE field of an SQLCA is the unmapped value as returned by the host or System i database server. The send and receive buffers are arranged from the oldest to the most recent within the file.

Each buffer contains the following information:

- ▶ The process ID
- ▶ A SEND BUFFER, RECEIVE BUFFER, or SQLCA label. The first DDM command or object in a buffer is labeled DSS TYPE.

The remaining data in send and receive buffers is divided into five columns, consisting of the following information:

- ▶ A byte count

- Example 8-9 shows the location of the db2drdat command in our AIX test server. db2drdat's location should be in your path so you do not need to look for where this tool is located.

```
# find . -name db2drdat
./opt/IBM/db2/V9.5/bin/db2drdat
./opt/IBM/db2/V9.5_01/bin/db2drdat
./opt/IBM/db2/V9.5 ESE/bin/db2drdat
```

### Example 8-10 db2drdat command syntax

```
>>-db2drdat----->
      .-----|
      .-on-. v   |
>-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-->
    +- -s-+     '- -l=length-'
    +- -c-+
    '- -i-'
-off-+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      '- -t---tracefile-' '- -p---pid-' '- -f-
```

- ▶ on  
This option turns on AS trace events (all if none specified).
- ▶ off  
This option turns off AS trace events.
- ▶ -r  
This option traces DRDA requests received from the DRDA AR.
- ▶ -s  
This option traces DRDA replies sent to the DRDA AR.
- ▶ -C  
This option traces the SQLCA received from the DRDA server on the host system. This is a formatted, easy-to-read version of not null SQLCAs.
- ▶ -i  
This option includes time stamps in the trace information.
- ▶ -l  
This option specifies the size of the buffer used to store the trace information.
- ▶ -p  
This option traces events only for this process. If -p is not specified, all agents with incoming DRDA connections on the server are traced. The pid to be traced can be found in the agent field returned by the LIST APPLICATIONS command.

► -t

This option specifies the destination for the trace. If a file name is specified without a complete path, missing information is taken from the current path. If tracefile is not specified, messages are directed to db2drdat.dmp in the current directory.

► -f

This option formats communications buffers.

If you do not specify a working directory, messages are directed to db2drdat.dmp in the current directory. In our test we are located in a temporary working directory and logged in as the DB2 Connect instance owner (db2inst1) on an Linux on z server. We executed the **db2drdat** command as shown in Example 8-11.

*Example 8-11 db2drdat on*

```
db2inst1@linux11:~> db2drdat on
Trace is turned on
db2inst1@linux11:~>
```

After having executed some DRDA activity, the trace was stopped as shown in Example 8-12.

*Example 8-12 Stopping the db2drdat traces*

```
db2inst1@linux11:~> db2drdat off
Trace truncated           : NO
Trace wrapped             : YES
Total number of trace records : 128
Number of trace records formatted : 128
Trace is turned off
db2inst1@linux11:~>
```

## DRDA trace output

For this execution, the file db2drdat.dmp is located in the current directory. Example 8-13 shows a partial output of a DRDA trace showing the connection from a DB2 Connect server to a DB2 9 for z/OS. Some of the data was removed for clarity.

*Example 8-13 DRDA trace example*

```
1 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
....
```

```
2 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
....
```

**SEND BUFFER(AR):**

	EXCSAT RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	00C9D041000100C3 1041007F115E8482	...A.....A...^..	.I}....C...".;db
...			

	ACCSEC RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0026D00100020020 106D000611A20003	.&..... .m.....	..}....._...s..
0010	00162110C4C2F9C1 4040404040404040	...!.....@@@@@@@@	....DB9A
0020	40404040404040	@@@@@@	

```
3 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
```

```
pid 7422 tid 2199066677168 cpid -1 node 0 sec 0 nsec 966875 probe 100
bytes 12
```

```
Data1 (PD_TYPE_UINT,4) unsigned integer:
158
```

- 4 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)  
 pid 7422 tid 2199066677168 cpid -1 node 0 sec 0 nsec 967718 probe 1178  
 bytes 175

#### RECEIVE BUFFER(AR):

```

          EXCSATRD OBJDSS          (ASCII)          (EBCDIC)
          0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF 0123456789ABCDEF
0000 008ED04300010088 14430044115EC4C2 ...C.....C.D.^.. ..}....h.....;DB
.....

```

---

In this example the DRDA trace is a materialization of the dialog between a requester and a server. Notice the text SEND BUFFER(AR) and RECEIVE BUFFER(AR) indicating the direction of the data flow.

### DRDA trace output file analysis

When reading a DRDA trace, you will find the DRDA commands like EXCSAT, ACCSEC and EXCSATRD as shown in this example. Refer to the The Open Group publication *DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture* for an explanation of the commands. The publication is available from the following Web page:

<http://www.opengroup.org/onlinepubs/9699939199/toc.pdf>

For instance, the command ACCSEC stands for Access Security. This command initializes the security mechanism and this publication provides the information required for understanding, for instance, the type of security mechanism used for this DRDA dialog.

Table 8-2 shows examples of DRDA commands.

Table 8-2 Some DRDA command

DRDA command	Purpose of command
EXCSAT	Exchange server attribute request
SECCHK	Security check
ACCRDB	Access relational DB
SECCHKRM	Security check complete reply message
EXCSATRD	Server attributes reply data
ACCRDBRM	Access relational DB reply message
PRPSQLSTT	Prepare SQL statement
OPNQRY	Open query request
SQLCARD	SQL communication reply data
OPNQRYRM	Open query complete reply message (with first block)
CNTQRY	Continue query (get next block or set of blocks)
QRYDTA	Query answer set data

DRDA command	Purpose of command
CLSQRY	Close query
ENDQRYRM	End of query reply message
RDBCMM	RDB commit unit of work
ENDUOWRM	End unit of work reply message

The following information is captured in a **db2drdat** trace:

- ▶ The process ID (PID) of the client application
- ▶ The RDB\_NAME cataloged in the database connection services (DCS) directory
- ▶ The DB2 Connect or Client CCSID(s)
- ▶ The host or System i database server CCSID(s)
- ▶ The host or System i database server management system with which the DB2 Connect system is communicating.

The first buffer contains the Exchange Server Attributes (EXCSAT) and Access RDB (ACCRDB) commands sent to the host or System i database server management system. It sends these commands as a result of a CONNECT TO database command. The next buffer contains the reply that DB2 Connect received from the host or System i database server management system. It contains an Exchange Server Attributes Reply Data (EXCSATRD) and an Access RDB Reply Message (ACCRDBRM).

### **EXCSAT**

The EXCSAT command contains the workstation name of the client specified by the Server Name (SRVNAM) object, which is code point X'116D', according to DDM specification. The EXCSAT command is found in the first buffer. Within the EXCSAT command, the values X'9481A292' (coded in CCSID 500) are translated to mask once the X'116D' is removed.

The EXCSAT command also contains the EXTNAM (External Name) object, which is often placed in diagnostic information about the host or System i database management system. It consists of a 20-byte application ID followed by an 8-byte process ID (or 4-byte process ID and 4-byte thread ID). It is represented by code point X'115E', and in this example its value is db2bp padded with blanks followed by 000C50CC. On a Linux or UNIX IBM Data Server Client, this value can be correlated with the **ps** command, which returns process status information about active processes to standard output.

### **ACCRDB**

The ACCRDB command contains the RDB\_NAME in the RDBNAM object, which is code point X'2110'. The ACCRDB command follows the EXCSAT command in the first buffer. Within the ACCRDB command, the values X'E2E3D3C5C3F1' are translated to STLEC1 once the X'2110' is removed. This corresponds to the target database name field in the DCS directory. The accounting string has code point X'2104'.

The code set configured for the DB2 Connect workstation is shown by locating the CCSID object CCSIDSBC (CCSID for single-byte characters) with code point X'119C' in the ACCRDB command. In this example, the CCSIDSBC is X'0333', which is 819.

The additional objects CCSIDDBC (CCSID for double-byte characters) and CCSIDMBC (CCSID for mixed-byte characters), with code points X'119D' and X'119E' respectively, are also present in the ACCRDB command. In this example, the CCSIDDBC is X'04B0', which is 1200, and the CCSIDMBC is X'0333', which is 819, respectively.

## EXCSATRD and ACCRDBRM

CCSID values are also returned from the host or System i database server in the Access RDB Reply Message (ACCRDBRM) within the second buffer. This buffer contains the EXCSATRD followed by the ACCRDBRM. The example output file contains two CCSID values for the host or System i database server system. The values are 1208 (for both single-byte and mixed byte characters) and 1200 (for double-byte characters).

If DB2 Connect does not recognize the code page coming back from the host or System i database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the host or System i database server does not recognize the code set sent from DB2 Connect, it will return VALNSPRM (Parameter Value Not Supported, with DDM code point X'1252'), which gets translated into SQLCODE -332 for the user.

The ACCRDBRM also contains the parameter PRDID (Product-specific Identifier, with code point X'112E').

## 8.1.4 JDBC traces

In this section we describe briefly the JDBC traces. For details refer to the *DB2 for Linux, UNIX, and Windows Information Center* at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>

First we look at the ways to activate Normal traces. At “db2trc – trace command” on page 341 we look at activating the service trace.

### Normal traces

If you have an SQLJ or JDBC application that is using the IBM Data Server Driver for JDBC and SQLJ, a JDBC trace can be enabled in different ways:

- ▶ If you use the DataSource interface to connect to a data source, use the DataSource.setTraceLevel() and DataSource.setTraceFile() method to enable tracing.
- ▶ If you use the DriverManager interface to connect to a data source, the easiest way to enable tracing is to set the logWriter on DriverManager before obtaining a connection. Example 8-14 shows how you can activate a trace using this method.

#### Example 8-14 Enabling traces using the DriverManager interface

```
DriverManager.setLogWriter(new PrintWriter(new FileOutputStream("trace.txt")));
```

- ▶ If you are using the DriverManager interface, you can alternatively specify the traceFile and traceLevel properties as part of the URL when you load the driver. This method is shown in Example 8-15 on page 338.

#### Example 8-15 Sample modification to connection string

```
String url =  
"jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A:traceFile=BSQLTrace.txt;traceLevel=" +  
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL + ";;";
```

The connection string has to specify the trace file name, trace level, and option for how detailed the trace needs to be. The available trace level options are as shown in Table 8-3.

Table 8-3 Trace level options

Trace level options	Information produced by the trace
TRACE_NONE	No traces are produced.

Trace level options	Information produced by the trace
TRACE_CONNECTION_CALLS	Gives information about AutoCommit value, create statement, commit, and close SQL statements.
TRACE_STATEMENT_CALLS	Gives information about the query execution, such as the SQL statement executed, and the time stamp the statement was returned.
TRACE_RESULT_SET_CALLS	Gives information about rows that were returned to the application when using a fetch.
TRACE_DRIVER_CONFIGURATION	Gives information about the client setup, such as operating system level, and information about the JRE version.
TRACE_CONNECTS	Gives information about the target DB, such as the version, the IP address, and port number.
TRACE_DRDA_FLOWS	Traces the DRDA data flow. The flow is shown in ASCII and EBCDIC.
TRACE_RESULT_SET_META_DATA	Gives information about the columns returned by the query as the column name, length, and data type.
TRACE_PARAMETERS_META_DATA	Describes input data received from the server.
TRACE_DIAGNOSTICS	Traces the SQLCA returned from the server.
TRACE_ALL	Gives all of the above trace information.

The following steps describe an alternative method for starting the trace:

1. Create a directory for trace output, for example /tmp/jdbc for UNIX.
2. In that directory create a properties file name, for example jcc.properties, with the following entries:
  - db2.jcc.traceDirectory=/tmp/jdbc
  - db2.jcc.traceFile=trace
  - db2.jcc.traceFileAppend=false
  - db2.jcc.traceLevel=-1
3. Add the following option to the command line that starts the JVM:

-Ddb2.jcc.propertiesFile=/tmp/jdbc/jcc.properties

You could also execute the program by executing the command:

```
java -Ddb2.jcc.propertiesFile=/tmp/jdbc/jcc.properties <ApplicationName>
```
4. Run the application. Trace output will go to the directory specified in the properties file.

Example 8-16 shows a sample JCC Type 4 trace. Different sections have been highlighted to show the information that is given by each trace level. This trace is the result of the command shown in Example 8-15 on page 338.

**Example 8-16 Sample JCC Type 4 trace file (partial output)**

```
[ibm][db2][jcc] [time:2009-04-21-15:52:04.417][thread:main][tracepoint:10] DataSource created. Table size: 1
[ibm][db2][jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] Driver: IBM DB2 JDBC Universal Driver Architecture 3.6.103
[ibm][db2][jcc] Compatible JRE versions: { 1.4, 1.5 }
[ibm][db2][jcc] Target server licensing restrictions: { z/OS: enabled; SQLDS: enabled; iSeries: enabled; DB2 for
Unix/Windows: enabled; Cloudscape: enabled }
[ibm][db2][jcc] Default fetch size: 64
[ibm][db2][jcc] Default isolation: 2
[ibm][db2][jcc] Collect performance statistics: false
```

```

[ibm][db2][jcc] No security manager detected.
[ibm][db2][jcc] Detected local client host: SVL-XXXXXXXX/1.11.111.1
[ibm][db2][jcc] Access to package sun.io is permitted by security manager.
[ibm][db2][jcc] JDBC 1 system property jdbc.drivers = null
[ibm][db2][jcc] Java Runtime Environment version 1.6.0_07
....
[ibm][db2][jcc] Dumping all file properties: { }
[ibm][db2][jcc] END TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] BEGIN TRACE_CONNECTS
[ibm][db2][jcc] Attempting connection to wtsc63.itso.ibm.com:12347/DB9A
[ibm][db2][jcc] Using properties: { kerberosServerPrincipal=null,
....
currentQueryOptimization=-2147483647, resultSetHoldability=0, readOnly=false,
supportsAsynchronousXARollback=2,securityMechanism=3 }
[ibm][db2][jcc] END TRACE_CONNECTS
[ibm][db2][jcc] [t4] [time:2009-04-21-15:52:04.678] [thread:main] [tracepoint:1] [Request.flush]
[ibm][db2][jcc][t4]          SEND BUFFER: EXCSAT          (ASCII)          (EBCDIC)
[ibm][db2][jcc][t4]          0 1 2 3 4 5 6 7   8 9 A B C D E F   0123456789ABCDEF 0123456789ABCDEF
[ibm][db2][jcc][t4] 0000   0098D04100010092   10410048115E8482   ...A.....A.H.^.. .q)....k.....;db
....

```

---

Example 8-17 shows a trace file extract for an SQLJ application.

*Example 8-17 Sample SQLJ trace file (partial output)*

```

[jcc] [Time:2009-04-15-10:47:24.390] [Thread:WebContainer : 0] [DB2ConnectionPoolDataSource@26e826e8] getPooledConnection
() called
[jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[jcc] Driver: IBM DB2 JDBC Universal Driver Architecture 3.53.70
[jcc] Compatible JRE versions: { 1.4, 1.5 }
[jcc] Target server licensing restrictions: { z/OS: enabled; SQLDS: enabled; iSeries: enabled; DB2 for Unix/Windows:
enabled; Cloudscape: enabled; Informix: enabled }
[jcc] Default fetch size: 64
[jcc] Default isolation: 2
[jcc] Collect performance statistics: false
[jcc] No security manager detected.
[jcc] Detected local client host: raxxxxxx/1.11.10.147
[jcc] Access to package sun.io is permitted by security manager.
[jcc] JDBC 1 system property jdbc.drivers = null
[jcc] Java Runtime Environment version 1.5.0
....
[jcc] Dumping all file properties: { }
[jcc] END TRACE_DRIVER_CONFIGURATION
[jcc] BEGIN TRACE_CONNECTS
[jcc] Attempting connection to wtsc63.itso.ibm.com:12347/DB9A
[jcc] Using properties: { maxStatements=0, currentPackagePath=NULLID,DEMO,.....
....
....currentDegree=null, DUMPMEM=null }
[jcc] END TRACE_CONNECTS
[jcc] [t4] [time:2009-04-15-10:47:24.760] [thread:WebContainer : 0] [tracepoint:315] creating a socket to
wtsc63.itso.ibm.com at 9.12.6.70
[jcc] [t4] [time:2009-04-15-10:47:25.191] [thread:WebContainer : 0] [tracepoint:1] [Request.flush]
[jcc] [t4]          SEND BUFFER: EXCSAT          (ASCII)          (EBCDIC)
[jcc] [t4]          0 1 2 3 4 5 6 7   8 9 A B C D E F   0123456789ABCDEF 0123456789ABCDEF
[jcc] [t4] 0000   0098D04100010092   10410048115E8482   ...A.....A.H.^.. .q)....k.....;db
...

```

---

The JDBC and SQLJ version and environment information can be found in Example 8-16 on page 339 and Example 8-17 on page 340 enclosed between the text **BEGIN TRACE\_DRIVER\_CONFIGURATION** and **END TRACE\_DRIVER\_CONFIGURATION**.

In both examples, you can identify the server to which the application is connected by referring to the information between the text **BEGIN TRACE\_CONNECTS** and **END TRACE\_CONNECTS**.



Example 8-18 shows a portion of a JCC Type 4 trace reporting an SQL Code error enclosed between the text `BEGIN TRACE_DIAGNOSTICS` and `END TRACE_DIAGNOSTICS`.

*Example 8-18 Sample JCC Type 4 trace file (partial output) showing SQL Error*

```
....
[ibm][db2][jcc] BEGIN TRACE_DIAGNOSTICS
[ibm][db2][jcc][Thread:main][SQLException@15f5897] java.sql.SQLException
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] DB2 SQLCA from server
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlCode      = -440
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlErrd      = { 1251846, 0, 0, -1, 0, 0 }
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlErrmc     = PROCEDURE;PAOLOR3.BSQL_ALONE
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlErrmcTokens = { PROCEDURE, PAOLOR3.BSQL_ALONE }
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlErrp      = DSNXOSTP
[ibm][db2][jcc][Thread:main][SQLException@15f5897][Sqlca@b162d5] SqlState     = 42884
....
[ibm][db2][jcc] END TRACE_DIAGNOSTICS
```

### db2trc – trace command

If more detailed tracing is required, you can use the **db2trc** command. This trace shows details of the entry, exit, and data that is passed from client to server. This trace is normally invoked when requested by IBM service support for detailed problem analysis and troubleshooting.

When **db2trc** is run, it invokes the trace and writes the trace information into the buffers. The trace should be invoked based on filter criteria, to avoid collecting a large amount of information. The **db2trc** can be invoked as follows:

1. From the client run the **db2trc on -m \*.\*.\*.\* -l 2000000 -t** command. This is shown in Example 8-19.

*Example 8-19 Starting db2trc*

```
db2inst1@linux11:~> db2trc on -m *.*.*.* -l 2000000 -t
Warning: The requested buffer size is not a power of 2. The buffer has
been rounded down to 1 megabytes.
Trace is turned on
```

Typing **db2trc** and pressing Enter lists the complete set of options that can be used. For details on these options refer to *IBM DB2 Database for Linux, UNIX, and Windows Information Center* at the following Web page:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>

In our example:

– -m

This refers to the mask that can be applied to trace specific events. In our case specifying the mask is the same as without the mask because the mask has requested all traces to be captured.

– -l

This refers to the last trace record being retained when buffers are full. The buffer size specified must be a power of 2, and must be a value between 1048576 and 134217728 bytes.

– -t

This refers to the time stamp being captured for references to events.

2. Run the application to reproduce the problem.

3. Once the application is run, run the **db2trc dump** command (as illustrated in Example 8-20) to get to the information captured by the trace.

*Example 8-20 db2trc dump command*

---

```
db2inst1@linux11:~> db2trc dump ./db2trc.dmp
Trace has been dumped to file "./db2trc.dmp"
```

---

In Example 8-20, ./db2trc.dmp is the trace output file name. The trace is not readable as it is. It needs to be formatted. But before proceeding to format, you must stop the existing trace.

4. To turn off the trace, use the **db2trc off** command illustrated in Example 8-21.

*Example 8-21 db2trc off command*

---

```
db2inst1@linux11:~> db2trc off
Trace is turned off
```

---

5. The trace file can be formatted using the command shown in the Example 8-22.

*Example 8-22 Format db2trc trace file command*

---

```
db2inst1@linux11:~> db2trc fmt ./db2trc.dmp ./db2trc.dmp.out
Trace truncated           : NO
Trace wrapped             : YES
Total number of trace records : 10999
Number of trace records formatted : 10999
```

---

A partial sample of a formatted report is shown in Example 8-23.

*Example 8-23 Sample formatted db2trc report*

---

```
1  entry DB2 UDB SQO Memory Management sqloGetPrivatePoolHandle fnc (1.3.129.7.0)
   pid 11134 tid 2199066045328 cpid -1 node -1 sec 0 nsec 0

2  exit DB2 UDB SQO Memory Management sqloGetPrivatePoolHandle fnc (2.3.129.7.0)
   pid 11134 tid 2199066045328 cpid -1 node -1 sec 0 nsec 265
   rc = 0
   bytes 16

   Data1 (PD_TYPE_PTR,8) Pointer:
   0x000002000505e6b8

3  mbt DB2 UDB SQO Memory Management sqlogmbkEx cei (6.3.129.36.2.32769)
   pid 11134 tid 2199066045328 cpid -1 node -1 sec 0 nsec 1296

   Marker:PD_OSS_ALLOCATED_MEMORY
   Description: Successfully allocated memory
   bytes 16

   Data1 (PD_TYPE_PTR,8) Pointer:
   0x000002000509e660

4  exit DB2 UDB SQO Memory Management sqlogmbkEx cei (2.3.129.36.2)
   pid 11134 tid 2199066045328 cpid -1 node -1 sec 0 nsec 1656
   codepath = 4:20:22

   rc = 0
   bytes 16

   Data1 (PD_TYPE_MEM_ADJUSTED_SIZE,8) Adjusted block size:
```

---

## 8.2 Accounting for distributed data with the EXCSQLSET command

Many enterprises need to implement resource monitoring practices, which allow system administrators to associate resource usage with individual user access. The results of these practices are used, for example, to charge individual users or their departments for the resources they consume or to do problem determination or auditing.

To enable an accounting or monitoring system to track DRDA access to a DB2 database server, a requester can send accounting and monitoring information using DRDA DDM commands to DB2 in one of two ways:

- ▶ Send an accounting identifier string in the prddta instance variable of the ACCRDB command with each application's connect request.
- ▶ Send accounting or monitoring identifier strings in the SQLSTT command data object of the EXCSQLSET command.

In this section, we explain how to use the EXCSQLSET command. DB2 server systems allow requester systems to influence certain accounting and monitoring information using the EXCSQLSET command. You can influence the following accounting information:

- ▶ End user IDs
- ▶ End user workstation names
- ▶ End user application names
- ▶ Accounting data

Much of this information is externalized in various forms in a DB2 system:

- ▶ The DSNV437I message of the DISPLAY THREAD command report
- ▶ THREAD-INFO data in various messages, such as DSNT375I
- ▶ The QWHC trace record correlation header
- ▶ The QMDA section of DB2 accounting trace records

To set the user ID, SQLSTT contains the string SET CLIENT USERID, followed by the user ID in single quotation marks. DB2 accepts a user ID of up to 16 characters and truncates any characters exceeding that length. See Example 8-24 on page 343.

*Example 8-24 SET CLIENT USERID example*

---

```
SET CLIENT USERID 'Toto'
```

---

To set the user workstation name, SQLSTT contains the string SET CLIENT WRKSTNNAME, followed by the workstation name in single quotation marks. DB2 accepts a name of up to 18 characters and truncates characters exceeding that length. This is illustrated in Example 8-25.

**Important:** The following examples are extracts of a DRDA trace, not to be confused with the SQL SET instructions. These registers cannot be directly set through SQL from DB2 for z/OS.

*Example 8-25 SET CLIENT WRKSTNNAME example*

---

```
SET CLIENT WRKSTNNAME 'TotoMac'
```

---

To set the user application name, SQLSTT contains the string SET CLIENT APPLNAME, followed by the application name in single quotation marks. DB2 accepts a name of up to 32 characters and truncates characters exceeding that length. This is shown in Example 8-26.

*Example 8-26 SET CLIENT APPLNAME example*

---

```
SET CLIENT APPLNAME 'TotoTestApplication'
```

---

To set the accounting information, SQLSTT contains the string SET CLIENT ACCTNG, followed by the accounting information in single quotation marks. DB2 accepts up to 255 characters and truncates characters exceeding that length. DB2 also assumes that the first 8 characters of accounting information are a product identifier (prdid). See Example 8-27.

*Example 8-27 SET CLIENT ACCTNG example*

---

```
SET CLIENT ACCTNG 'TotoAccountingInfo'
```

---

This information provides you with a better understanding of the accounting information, and you may use these fields for WLM workload classification. Example 8-28 shows an extract of WLM classification rules where we classify DDF workload using the accounting information set using this method. WLM will classify a DDF workload using any TotoA\* in SET CLIENT ACCTNG into the Service Class DDFTOT.

*Example 8-28 Using accounting information for DDF work classification*

---

\* Subsystem Type DDF - DDF Work Requests

Last updated by user PAOLOR6 on 2009/04/22 at 20:38:30

Classification:

Default service class is DDFBAT  
There is no default report class.

Qualifier # type	Qualifier name	Starting position	Service Class	Report Class
-----				
1 SI	DB9A		DDFDEF	RDB9ADEF
2 . PC	. TRX*		DDFONL	RSSL
<b>2 . AI</b>	<b>. TotoA*</b>	<b>56</b>	<b>DDFTOT</b>	
2 . UI	. PAOLOR3			RNISANTI
...				

---

Note the value in the Starting position column of the classification rules shown in this example. The accounting information evaluation starts at columns 56 because the data entered in the SET CLIENT ACCTNG instruction is not added at the beginning of the Accounting Information field, as indicated in Example 8-29.

*Example 8-29 Enclave details showing the Accounting Information field*

---

RMF Enclave Classification Data

---

[illegible]

```

Subsystem Type: DDF      Owner: DB9ADIST      System: SC63
Accounting Information . . :
SQL09050NT              TotoTestApplication Toto      TotoAc
countingInfo

```

<http://publib.boulder.ibm.com/infocenter/zos/v1r10/index.jsp?topic=/com.ibm.zos.r10.erb500/erbzra8059.htm>

RMF V1R10 Enclave Report					Line 1 of 3						
Command ==>					Scroll ==> CSR						
Samples: 60		System: SC63		Date: 04/29/09	Time: 14.32.00	Range: 60		Se			
Current options:		Subsystem Type: ALL				-- CPU Util --					
		Enclave Owner:				Appl%		EApp%			
		Class/Group:				1.4		2.2			
Enclave	Attribute	CLS/GRP	P	Goal	% D X	EApp%	TCPU	USG	DLY	IDL	
*SUMMARY						0.004					
ENC00001		DDFTOT	1	0.500	80	0.003	0.081	17	0.0	0.0	
ENC00002		DDFONL	2		40	0.001	9.733	0.0	0.0	0.0	

You can set these parameters in many ways, including through the Configuration Assistant, as shown in Figure 8-8.

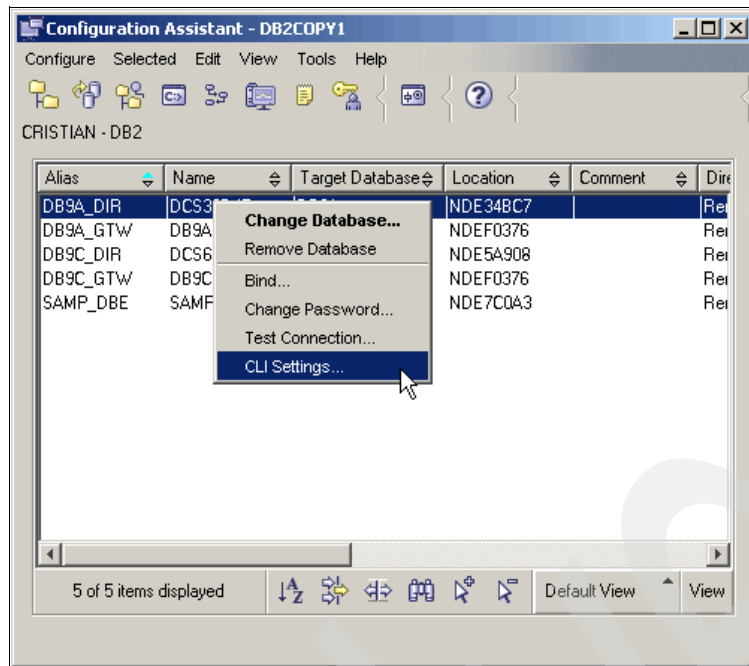


Figure 8-8 Changing the CLI setting using the DB2 Configuration Assistant

Selecting the CLI Setting option, you can ADD the CLI settings that activate the functions described in this section. The following settings were set for the purpose of this test:

- ▶ ClientApplName = TotoTestApplication
- ▶ ClientWrkstnName = TotoMac
- ▶ Clientuser ID = Toto
- ▶ ClientAcctStr = TotoAccountingInfo

These settings are part of the Transaction section of the CLI settings. See Figure 8-9.

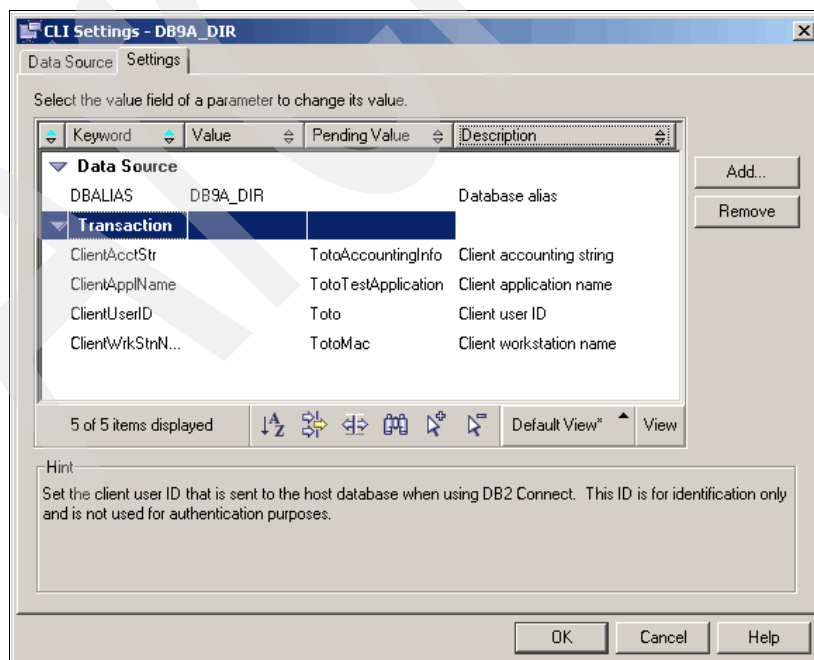


Figure 8-9 CLI settings: Transaction section

Another way of checking these settings, in case you do not have access to the Configuration Assistant, is to execute the command **get cli cfg** as shown in Example 8-31. This command is described in Chapter 7, “Performance analysis” on page 269. The example shows also that these definitions are by database. This client had cataloged several databases, but CLI accounting settings were done only for database DB9A\_DIR.

*Example 8-31 db2 get cli cfg and accounting CLI settings*

---

```
C:\Program Files\IBM\SQLLIB\BIN>db2 get cli cfg
```

```
Section: DB9C_GTW
```

```
-----
DBALIAS=DB9C_GTW
```

```
Section: DB9C
```

```
-----
DBALIAS=DB9C
```

```
Section: DB9A_DIR
```

```
-----
ClientAcctStr=TotoAccountingInfo
ClientUserID=Toto
DBALIAS=DB9A_DIR
ClientWrkStnName=TotoMac
ClientApplName=TotoTestApplication
UID=paolor4
```

```
C:\Program Files\IBM\SQLLIB\BIN>
```

---

APAR PK74330 introduces the stored procedure SYSPROC.WLM\_SET\_CLIENT\_INFO in DB2 for z/OS V8 and 9. This procedure allows the caller (DB2 for z/OS Application Requester) to set client information associated with the current connection at the DB2 for z/OS server, changing the following DB2 for z/OS client special registers:

- ▶ CURRENT CLIENT\_ACCTNG
- ▶ CURRENT CLIENT\_USERID
- ▶ CURRENT CLIENT\_WRKSTNNAME
- ▶ CURRENT CLIENT\_APPLNAME

The existing behavior of the CLIENT\_ACCTNG register is unchanged. It will continue to get its value from the accounting token for DSN requesters, and from the accounting string for SQL and other requesters. In DB2 V8, SYSPROC.WLM\_SET\_CLIENT\_INFO requires new-function mode.

Example 8-32 is a snippet of a network trace report showing DRDA traffic between the client and DB2 for z/OS. The intent of this example is to show how the CLI settings described on this section are translated by the driver on SQL Statements. You can see in this example how the EXCSQLSET command described above, is sent to the server.

You also can find the DRDA Distributed Data Management (DDM) Architecture command SQLSTT, SQL Statement. It specifies that an SQL statement is being either executed or bound into a package. The SQL language standard or product unique extensions, not the DDM architecture, define the syntax and semantics of the SQL statement.

*Example 8-32 DRDA network trace and CLI accounting settings*

---

```
DRDA (Set SQL Environment)
```

```

DDM (EXCSQLSET)
...
DRDA (SQL Statement)
DDM (SQLSTT)
....
SQL statement (ASCII): \030SET CLIENT USERID 'Toto'
SQL statement (EBCDIC): \030... .<..+. .... '?.?'.
DRDA (SQL Statement)
DDM (SQLSTT)
....
SQL statement (ASCII): \037SET CLIENT WRKSTNNAME 'TotoMac'
SQL statement (EBCDIC): \037... .<..+. ....++.(. '?.?(/.''.
DRDA (SQL Statement)
DDM (SQLSTT)
....
SQL statement (ASCII): )SET CLIENT APPLNAME 'TotoTestApplication'
SQL statement (EBCDIC): )... .<..+. .&&<+. (. '?.?.....%../..?>'.
DRDA (SQL Statement)
DDM (SQLSTT)
....
SQL statement (ASCII): eSET CLIENT ACCTNG 'SQL09050NT
TotoTestApplication Toto ',X'12','TotoAccountingInfo'
SQL statement (EBCDIC): .... .<..+. ....+. '...<.9.5.+
.?.?.....%../..?> .?.? ', '.2','.?.?...?>..>..>?'.

```

---

Example 8-33 shows two threads originated from the same workstation and using the same version of DB2 Control Center but two differently cataloged databases against the same DB2 fro z/OS subsystem. Compare both information and appreciate how the accounting information is reflected. Notice that the AUTHID data is the same for both threads.

*Example 8-33 DIS THD and accounting information*

```

DSNV401I -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I -DB9A ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID    PLAN      ASID TOKEN
SERVER    RA *   35 javaw.exe    PAOLOR4   DISTSERV 00A8 456
V437-WORKSTATION=CRISTIAN, USERID=pao4or4,
APPLICATION NAME=javaw.exe
V442-CRTKN=172.16.209.128.55557.0904011742
V445-G90C0646.J03B.C3F922218D6B=456 ACCESSING DATA FOR
( 1)::9.30.28.113
V447--INDEX SESSID          A ST TIME
V448--( 1) 12347:64228      W R2 0909113424764
SERVER    RA *   41 javaw.exe    PAOLOR4   DISTSERV 00A8 451
V437-WORKSTATION=TotoMac, USERID=Toto,
APPLICATION NAME=TotoTestApplication
V441-ACCOUNTING=TotoAccountingInfo
V442-CRTKN=172.16.209.128.54789.0904011742
V445-G90C0646.J03B.C3F921F55CD8=451 ACCESSING DATA FOR
( 1)::9.30.28.113
V447--INDEX SESSID          A ST TIME
V448--( 1) 12347:64224      W R2 0909113421530

```

---

Figure 8-10 is an example of OMEGAMON PE panel from which you can see how this information is reflected in the thread details.



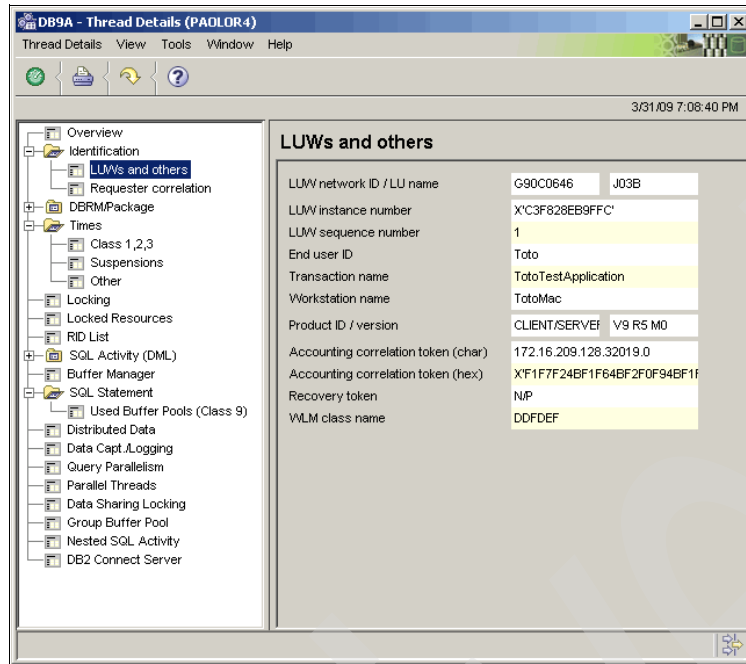


Figure 8-10 Accounting information in OMEGAMON PE

To further illustrate the advantages of setting the user information, consider Example 8-34. It shows a timeout report before activating the CLI settings.

Example 8-34 Timeout report when not using CLI user information

```

13.51.46 STC27893 DSNT376I -DB9A PLAN=DISTSERV WITH 373
373 CORRELATION-ID=javaw.exe
373 CONNECTION-ID=SERVER
373 LUW-ID=G90C0646.J03B.C3F923DB92E0=477
373 THREAD-INFO=PAOLOR4:CRISTIAN:paolor4:javaw.exe
373 IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS
PLAN=DSNESP RR
373 WITH
373 CORRELATION-ID=PAOLOR4
373 CONNECTION-ID=TSO
373 LUW-ID=USIBMSC.SCPDB9A.C3F923D3CAE5=476
373 THREAD-INFO=PAOLOR4:*.~.*
373 ON MEMBER DB9A
13.51.46 STC27893 DSNT501I -DB9A DSNILMCL RESOURCE UNAVAILABLE 374
374 CORRELATION-ID=javaw.exe
374 CONNECTION-ID=SERVER
374 LUW-ID=G90C0646.J03B.C3F923DB92E0=477
374 REASON 00C9008E
374 TYPE 00000210
374 NAME DB2PEDB .DB2PMFAC.00000001

```

The timeout report for the same SQL operation is shown in Example 8-35 after the activation of CLI settings. This message now provides more details and it helps in identifying the applications involved in the timeout.

*Example 8-35 Timeout report when using CLI custom information*

---

```
14.01.42 STC27893 DSNT376I -DB9A PLAN=DISTSERV WITH 378
378 CORRELATION-ID=javaw.exe
378 CONNECTION-ID=SERVER
378 LUW-ID=G90C0646.J03B.C3F92617F347=484
378 THREAD-INFO=PAOLOR4:TotoMac:Toto:TotoTestApplication
378 IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS
PLAN=DSNESP RR
378 WITH
378 CORRELATION-ID=PAOLOR4
378 CONNECTION-ID=TSO
378 LUW-ID=USIBMSC.SCPDB9A.C3F92607F056=483
378 THREAD-INFO=PAOLOR4:*:*:*
378 ON MEMBER DB9A
14.01.42 STC27893 DSNT501I -DB9A DSNILMCL RESOURCE UNAVAILABLE 379
379 CORRELATION-ID=javaw.exe
379 CONNECTION-ID=SERVER
379 LUW-ID=G90C0646.J03B.C3F92617F347=484
379 REASON 00C9008E
379 TYPE 00000210
379 NAME DB2PEDB.DB2PMFAC.00000001
```

---

## 8.2.1 TCP/IP packet tracing on z/OS

To determine the network component of the elapsed time of a transaction, you need to use a TCP/IP packet trace. There are two main component traces in TCP/IP on z/OS:

- ▶ SYSTCPIP(CTRACE)  
CTRACE identifies the components and is helpful in debugging at the component level.
- ▶ SYSTCPDA(PKTTRACE)  
SYSTCPDA enables a packet trace and gives information about the connection times.

Refer to the IBM publication *z/OS V1R10.0 Communication Server: IP Diagnosis Guide*, GC31-8782-09, for details about the steps described in this section.

The next sections show how to generate and analyze the packet trace.

### Generating the packet trace

The following steps are needed to set up the TCP/IP packet trace.

1. To have a packet trace written to an external data set, a started task needs to be created (for example, in SYS1.PROCLIB), so that the trace can be captured. Example 8-36 shows a sample procedure.

*Example 8-36 Sample procedure for capturing CTRACE information*

---

```
//CTWTR1 PROC
/* PROC USED FOR CAPTURING CTRACE OUTPUT TO AN EXTERNAL */
/* WRITER ... CHANGE THE OUTPUT DATA SETS AS REQUIRED TO */
/* A MAX OF 16 */
/*
//IEFPROC EXEC PGM=ITTTTCWR
//TRCOUT01 DD DSN=BARTR1.TEST.CTRACE.PING.NONHIPER,UNIT=3390,
// SPACE=(CYL,(100),,CONTIG),DISP=(NEW,CATLG)
```

---

2. Start the external writer with the following command so that the packet trace information can be captured in the data set:

```
TRACE CT,WTRSTART=CTWTR1
```

CTWTR1 is the procedure name created in step 1. Alternatively, the CTRACE can be started such that it writes the information in to the TCP/IP data space. The same data set can be used for both the packet trace (SYSTCPDA) and CTRACE (SYSTCPIP).

3. Start the SYSTCPDA component trace with the following command:

```
TRACE CT,ON,COMP=SYSTCPDA,SUB=(TCPIPA)
R 65,WTR=CTWTR1,END
```

The required reply attaches the external writer that was previously started, so that it can be used for storing the information. In the above command TCPIPA is the TCP/IP procedure name on our system.

4. Verify whether the traces have been started properly, using the following command:

```
D TRACE,COMP=SYSTCPDA,SUB=(TCPIPA)
```

The output of the command is shown in Example 8-37 on page 351.

*Example 8-37 Output of display trace command*

---

```
D TRACE,COMP=SYSTCPDA,SUB=(TCPIPA)
IEE843I 18.50.43 TRACE DISPLAY 831
      SYSTEM STATUS INFORMATION
ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
=====
SYSTCPDA
      MODE BUFFER HEAD SUBS
      =====
      OFF          HEAD    2
      NO HEAD OPTIONS
SUBTRACE  MODE BUFFER HEAD SUBS
-----
TCPIPA    ON    0016M
ASIDS     *NONE*
JOBNAMES  *NONE*
OPTIONS   MINIMUM
WRITER    CTWTR1
```

---

5. Starts the packet trace processing for TCP/IP with the following command:

```
V TCPIP,TCPIPA,PKT,ON,IP=9.12.6.8
```

The step requires that you specify the IP address of the machine that you are monitoring. (In an n-tier context, it is the DB2 Connect Server or the DB2 client that is connecting to the mainframe server.) If the IP address is not specified, then all traffic will be traced.

Now it is time to simulate the problem so that the packet trace can be captured.

6. Issue the following command to stop the packet trace once the event is complete:

```
V TCPIP,TCPIPA,PKT,OFF
```

7. Disconnect the external writer with the following command:

```
TRACE CT,ON,COMP=SYSTCPDA,SUB=(TCPIPA)
R 70,WTR=DISCONNECT,END
```

8. The external writer is stopped, so that the trace data set can be used for further analysis by using the following command:

```
TRACE CT,OFF,COMP=SYSTCPDA,SUB=(TCPIPA)
TRACE CT,WTRSTOP=CTWTR1,FLUSH
```

## Analyzing the packet trace

Once the trace is written (to check, you can verify the 'percent used' of the trace data set information), use IPCS to format the trace and get the required information about the network times. We assume that your systems programmer has set up the IPCS environment for you, such as allocating a so-called dump directory. On the IPCS primary option menu select option **2.7.1** to go to the CTRACE option menu as in Example 8-38.

*Example 8-38 IPCS primary options menu*

```
----- z/OS 01.04.00 IPCS PRIMARY OPTION MENU -----
OPTION ==> 2.7.1

0 DEFAULTS      - Specify default dump and options
1 BROWSE        - Browse dump data set
2 ANALYSIS      - Analyze dump contents
3 UTILITY        - Perform utility functions
4 INVENTORY      - Inventory of problem data
5 SUBMIT         - Submit problem analysis job to batch
6 COMMAND        - Enter subcommand, CLIST or REXX exec
T TUTORIAL       - Learn how to use the IPCS dialog
X EXIT          - Terminate using log and list defaults

*****
* USERID   - BART
* DATE     - 03/05/30
* JULIAN    - 03.150
* TIME     - 17:02
* PREFIX   - BART
* TERMINAL - 3278A
* PF KEYS  - 24
*****
```

Enter END command to terminate IPCS dialog

This takes you to the panel shown in Example 8-39.

*Example 8-39 PCS CTRACE option menu*

```
----- CTRACE OPTION MENU -----
OPTION ==> D

To display CTRACE information, enter the corresponding option.

Q QUERY      - Specify parameters for QUERY report
D DISPLAY     - Specify parameters to display CTRACE entries
S START      - Start CTRACE subcommand as specified below
R RESET      - Reset the CTRACE parameters
```

Pending CTRACE subcommand shown below:

END/PF3 = terminate CTRACE processing.

Select option **D** to specify the report options (Example 8-40). Make sure to specify the correct component trace type (SYSTCPDA). Specify the data set name of the CTRACE that you want to format in the 'override source' field. (This overrides IPCS's default input data set.) As report type, you can specify TALLY. That is the simplest report. You can specify additional SYSTCPDA reporting options in the 'options' field.

Use the **S** command (on the command line), to start formatting.

#### Example 8-40 PCS CTRACE reporting options

```
----- CTRACE DISPLAY PARAMETERS -----
COMMAND ==> S

System      ==>          (System name or blank)
Component   ==> SYSTCPDA (Component name (required))
Subnames    ==>

GMT/LOCAL   ==> L          (G or L, GMT is default)
Start time  ==>          (mm/dd/yy, hh:mm:ss.ddd or
Stop time   ==>          mm/dd/yy, hh:mm:ss.ddd)
Limit       ==> 0          Exception ==>
Report type ==> TALLY      (Short, Summary, Full, Tally)
User exit   ==>          (Exit program name)
Override source ==> DSNAME('BARTR1.CTRACE.ADHOC.BLK.NHIPR')
Options     ==>

To enter/verify required values, type any character
Entry IDs ==>  Jobnames ==>  ASIDs ==>  OPTIONS ==>  SUBS ==>

CTRACE COMP(SYSTCPDA) LOCAL TALLY

ENTER = update CTRACE definition.  END/PF3 = return to previous panel.
S = start CTRACE.  R = reset all fields.
```

If this is the first time you format this trace data set, you will get the set of messages shown in Example 8-41. Reply Y, as this is a trace data set.

#### Example 8-41 Trace initialization message

```
TIME-06:04:11 PM. CPU-00:00:03 SERVICE-84525 SESSION-04:26:06 MAY 30,2003
Initialization in progress for DSNAME('BARTR1.CTRACE.ADHOC.BLK.NHIPR')
Treat input only as trace data? Enter Y for yes, N for full initialization
Y
```

A snippet of the formatted output is shown in Example 8-42.

#### Example 8-42 Sample formatted trace output

```
IPCS OUTPUT STREAM -----
Command ==>
***** TOP OF DATA *****

COMPONENT TRACE TALLY REPORT
SYSNAME(SC63)
COMP(SYSTCPDA)
z/OS TCP/IP Packet Trace Formatter, (C) IBM
DSNAME('BARTR1.CTRACE.ADHOC.BLK.NHIPR')

No packets required reassembly

=====
SYSTCPDA Trace Statistics
11,456 ctrace records processed
```

```

124 segmented trace records read
0 segmented trace records were lost
11,394 trace records read
0 records could not be validated
11,394 records passed filtering
11,394 packet trace records processed
0 data trace records processed
=====

```

When using IPCS to format CTRACE data, SYSTCPDA formatting allows you to export the packet trace data to a data set in comma delimited format. To do so perform the following steps:

1. Allocate a data set and associate it with the DDNAME export. Use the following command on the CTRACE panel to do this:

```
TSO ALLOC DA(TEST.EXPORT) DDNAME(EXPORT)
```

2. Indicate that the report generator has to use this data set by specifying export in the options field on the CTRACE panel, or directly issue the following command:

```
CTRACE COMP(SYSTCPDA)SUMMARY OPTIONS((SESSION, EXPORT)) DSNAME('input.trc.dsn')
```

When the EXPORT option is used, the output is written to the data set allocated to the EXPORT DDNAME in comma separated value (CSV), for example. (The export data set name is not specified in the command.) You can then use the data set for further processing, for example, using REXX or a spreadsheet to summarize the results.

In the command above we use the SUMMARY option. You can also use the TALLY option. It does not affect the output of the export data set.

In the command above, the SESSION option is also used. If multiple sessions are captured, for example, because the trace did not specify an IP mask, the SESSION option will break the output into multiple sessions and display information per session, which is more convenient for analysis. The output is shown in Example 8-43.

*Example 8-43 Formatted report from IPCS for packet trace*

Client or Server:	SERVER,	CLIENT
Port:	33730,	1413
Application:	,	
Link speed (parm):	10,	10 Megabits/s
Connection:		
First timestamp:	2003/03/03 19:38:21.618849	
Last timestamp:	2003/03/03 19:38:29.068710	
Duration:	00:00:07.449861	
Average Round-Trip-Time:		0.012 sec
Final Round-Trip-Time:		0.122 sec
Final state:	CLOSED (PASSIVE CLOSE)	
Out-of-order timestamps:		0
Data Quantity & Throughput:		
Application data bytes:	Inbound,	Outbound
Sequence number delta:	7055,	8542
Total bytes Sent:	7057,	8543
Bytes retransmitted:	7147,	8542
Throughput:	92,	0
	1.183,	1.433 Kilobytes/s

Bandwidth utilization:	0.09%,	0.11%
Delay ACK Threshold:	200,	200 ms
Minimum Ack Time:	0.000318,	0.000076
Average Ack Time:	0.021007,	0.000098

---

The report shows the time taken to execute the transaction at the server and the time taken at the client. If the accounting report is available, then the network time can be calculated.

Example 8-44 summarizes the time spent at the client and at the server, which can also be obtained using the SESSION option in the IPCS. The other relevant information that can be used for tuning is the “maximum time to next client request/server response” field. This field indicates if any of the processes has taken a long time at the client before the client or server responded. It also provides the record number in the trace, which can help to trace back the problem in the application.

*Example 8-44 Summary of the IP trace CSV format file using the inhouse tool*

---

```
Total Session Elapsed Time = 53.269337
Total Session TCP Data Traced = 49960
Total Number of Packets From Client = 306
Total TCP Data from Client = 33934
Maximum TCP Size from Client = 436
Elapsed Time from Client = 52.817882
Number of Client Requests = 151
Average Time/Request from Client = 0.349787298013
Minimum Time to Next Client Request = 0.001327
Maximum Time to Next Client Request = 0.519695 at record 29
Total Number of Responses From Server = 154
Total TCP Data from Server = 16026
Maximum TCP Size from Server = 1440
Elapsed Time from Server = 0.451455
Number of Server Responses = 152
Average Time/Response from Server = 0.00298976821192
Minimum Time to Next Server Response = 0.000170
Maximum Time to Next Server Response = 0.015283 at record 465
```

---

## 8.3 Network analyzer

In an n-tier environment there can be many servers between the client and the database server. The messages from the client are passed to the server and back over the interconnecting networks. With many servers and the messaging pattern, the network time can be a major time consumer in the user response time. When it comes to problem determination, tools such as network analyzers are able to give a better picture of the traffic at the physical layer.

Network analyzers can give a complete picture of the traffic flowing out and coming into the machine (client/server) being monitored. Network analyzers come in many types. The network analyzer can be hardware or software. The analyzers can range from sophisticated devices to simple software-based analyzers that are able to perform some basic functionality.

Using a network analyzer may be one of the best ways of understanding and monitoring the DRDA protocol.

In this publication, to study the time spent in the network component of the n-tier architecture, a freeware network analyzer software called Wireshark is used. Getting started with

Wireshark is straightforward. You can download the installation software from the following Web page:

<http://www.wireshark.org>

The version used on this book is 1.0.6. Wireshark is available under the GNU General Public License version 2. Wireshark supports formatting the DRDA protocol. For details of the supported fields, refer to the following Web page:

<http://www.wireshark.org/docs/dfref/d/drda.html>

Figure 8-11 on page 356 describe the options we used during the tests shown in this book. You can access this panel by navigating to **Capture** → **Options** from the main panel.

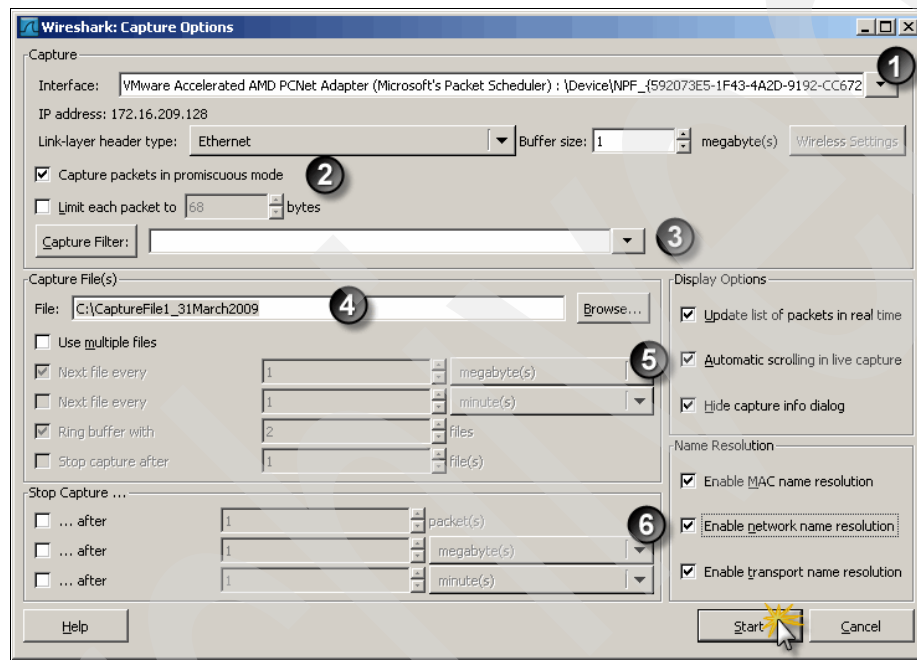


Figure 8-11 Network analyzer options panel

The numbers shown in Figure 8-11 are explained as follows:

1. Select the interface on which the monitoring will be done. It is necessary to select the physical or virtual adapter through which the DRDA traffic will occur.
2. Select the “Capture in promiscuous mode” check box to monitor all the network traffic that passes through the adapter. If you do not, only the traffic sent from or received to the network adapter will be monitored. Not working in promiscuous mode may reduce considerably the size of the trace capture.
3. Filter the address or port or protocol to be monitored in the “Capture filter” text box.
4. Identify a file where the monitor trace will be stored.
5. Select the “Automatic scrolling in live capture” check box for easier reading of the trace information.
6. Select the “Enable network name resolution” check box so the tool displays dns entries instead of IP addresses, when possible.
7. Click **Start** to start capturing data.



Figure 8-12 shows Wireshark's main panel during a capture session.

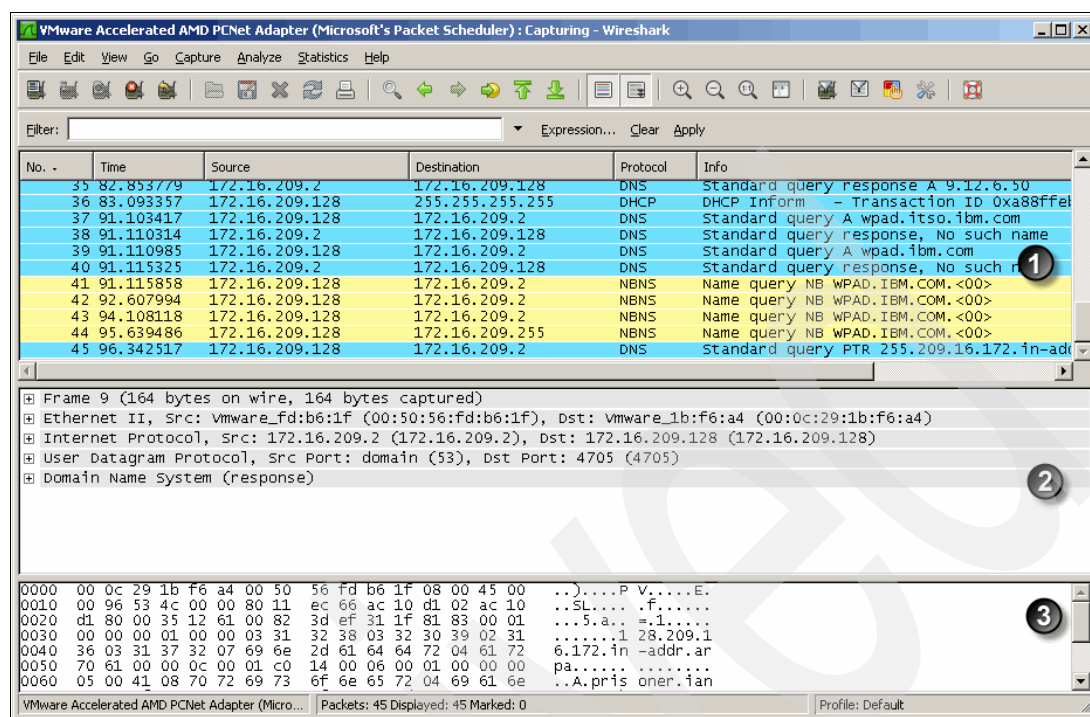


Figure 8-12 Network analyzer main panel

The numbers in Figure 8-12 are explained as follows:

1. Packets panel. This section shows the network packets as they are collected. Source, target, protocol and other information are available.
2. Packets details. By selecting a packet in the packets panels, you get detailed information about this section.
3. Raw packet data.

Being comfortable with the use of this tool can be useful for problem determination. Consider the following example. If you encounter a problem that you suspect is related to security settings, you may execute a monitored test and analyze the DRDA packets.

Refer to Figure 8-13 on page 358 for an example of a client to server connection monitored by Wireshark. By selecting the SECMEC DDM code point, you can identify the security mechanism being used. SECMEC is described in the DRDA DDM reference. When SECMEC flows from the source server to the target server, the SECMEC parameter specifies the security mechanism combination that the source server wants to use. When SECMEC flows from the target server to the source server, the SECMEC parameter must either reflect the value requested by the source server or if the target server does not support the requested security mechanism, then the target server returns the SECMEC values that it does support.

For this example, the value being transferred is x'0003', which identifies that user ID and password will be used as security mechanism.

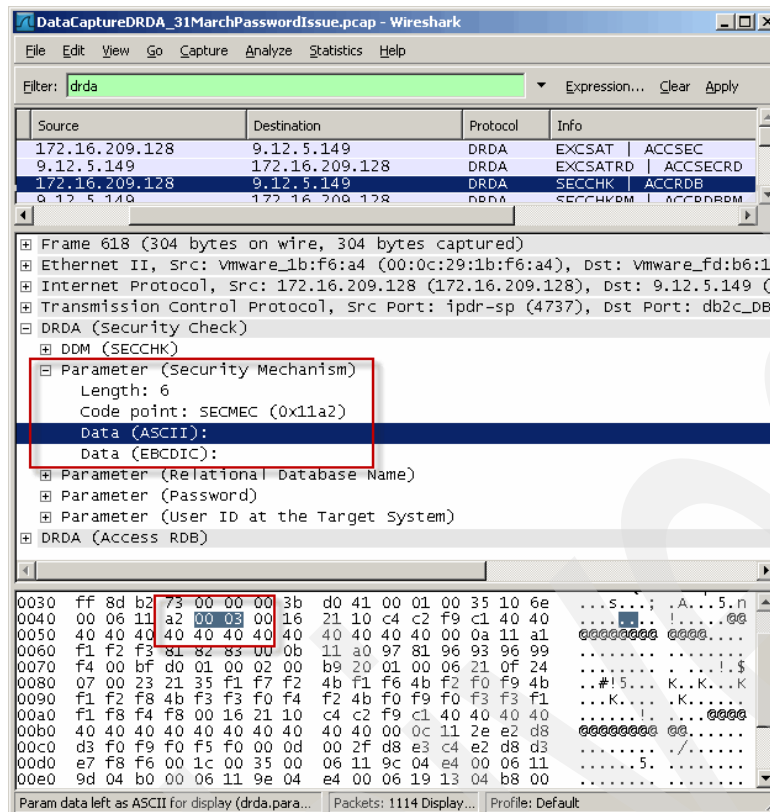


Figure 8-13 DRDA SECMEC DDM code point monitored by Wireshark

The network monitoring can create huge amounts of data depending on the activity. Wireshark provides the capability of filtering packets. The data transfer between an application and DB2 for z/OS uses not only DRDA packets, but also TCP. Filtering on DRDA only packets may hide information of interest on problem determination. This can be done by entering the word `drda` in the filter option in the main panel, as shown in Figure 8-14.

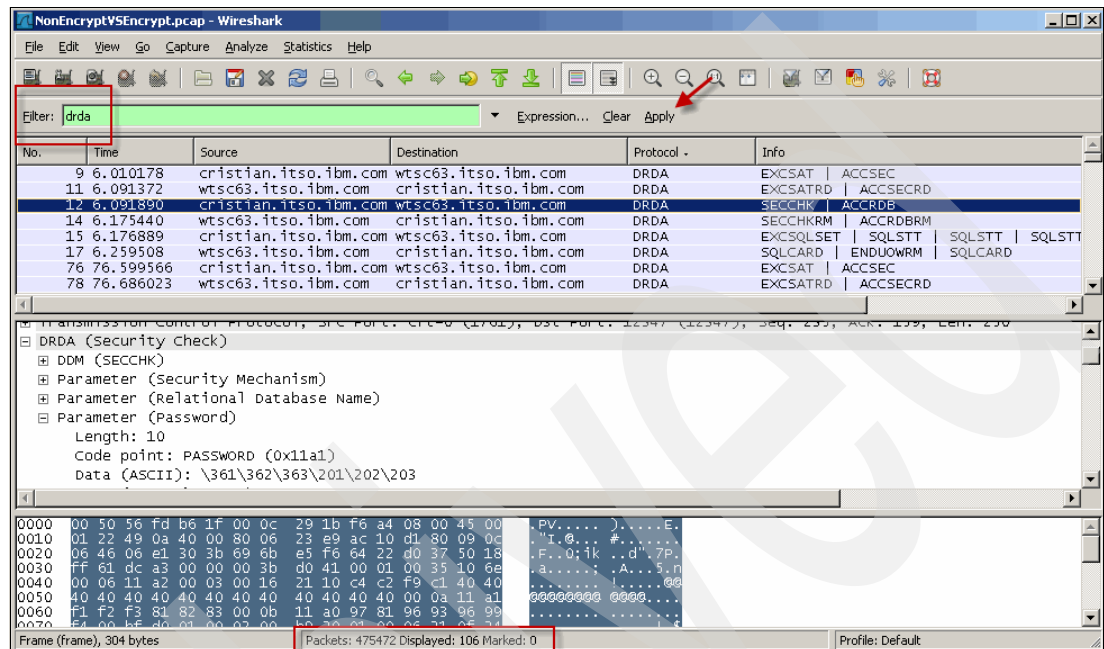


Figure 8-14 Apply DRDA protocol filter to Wireshark capture

Further filtering can be done by, for example, specifying an IP address and / or a port. An example is shown in Figure 8-15. For details on filtering capabilities of Wireshark, refer to the following Web page:

<http://www.wireshark.org>

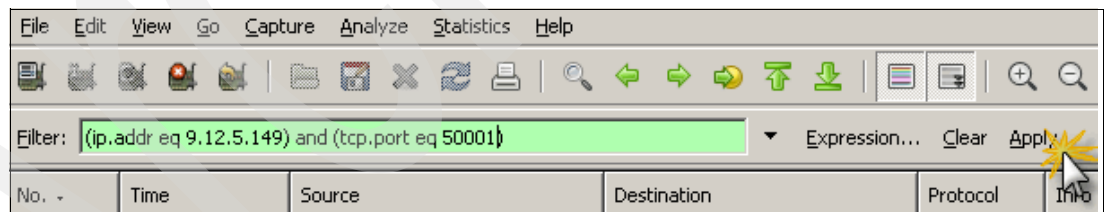


Figure 8-15 Filtering packets in Wireshark

You can export selected data to a comma separated value (csv) file to have it manipulated in a spreadsheet. You can access this function from the menu **File → Export → File**. Figure 8-16 shows an example where the DRDA packets involved in the authentication of a client are isolated and ordered by execution sequence. Two different colors identify two different tests. Send and receive packets can be identified by the source and target columns.

Microsoft Excel - DRDA_Encrypt_vs_NonEncrypt_csv_Sample						
E137						
	A	B	C	D	E	F
1	Nbr.	Time	Source	Target	Protocol	Step
2	9	6.010178	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	1
3	11	6.091372	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	2
4	12	6.091890	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	3
5	14	6.175440	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	4
6	15	6.176889	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	5
7	17	6.259508	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	6
8	76	76.599566	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	1
9	78	76.686023	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	2
10	79	76.716251	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	3
11	81	76.804820	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	4
12	82	76.806407	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	5
13	84	76.887759	wtsc63.itso.ibm.com	cristian.itso.ibm.com	DRDA	6
132						
133						
DRDA_Encrypt_vs_NonEncrypt_csv_						
Ready						

Figure 8-16 Analysis of DRDA packets in a spreadsheet

You can also export the packets to a file for analysis. Part of the file for the DRDA commands shown in the previous figure are shown in Example 8-45. You can identify in this trace the source and destination servers and part of the DRDA steps for security check. The target database (DB9A) is identified by the DDM code point RDBNAM.

Example 8-45 Analysis of DRDA packets in a flat file

No.	Time	Source	Destination	Protocol	Info
3	0.081712	cristian.itso.ibm.com	wtsc63.itso.ibm.com	DRDA	SECCHK   ACCRDB
Frame 3 (304 bytes on wire, 304 bytes captured)					
...					
Internet Protocol, Src: cristian.itso.ibm.com (172.16.209.128), Dst: wtsc63.itso.ibm.com (9.12.6.70)					
...					
Source: cristian.itso.ibm.com (172.16.209.128)					
Destination: wtsc63.itso.ibm.com (9.12.6.70)					
Transmission Control Protocol, Src Port: cft-0 (1761), Dst Port: 12347 (12347), Seq: 235, ...					
DRDA (Security Check)					
DDM (SECCHK)					
Length: 59					
Magic: 0xd0					
Format: 0x41					
0... = Reserved: Not set					
.1.. = Chained: Set					
..0. = Continue: Not set					
...0 = Same correlation: Not set					
DSS type: RQSDSS (1)					
CorrelId: 1					
Length2: 53					
Code point: SECCHK (0x106e)					
Parameter (Security Mechanism)					
Length: 6					
Code point: SECMEC (0x11a2)					
Data (ASCII):					
Data (EBCDIC):					
Parameter (Relational Database Name)					
Length: 22					
Code point: RDBNAM (0x2110)					
Data (ASCII): \304\302\371\301@@@@@@@@@@@@					

As an example of utilization, consider a DB2 connection done from a Windows workstation where DB2 Enterprise Server Edition (ESE) was installed and a DB2 9 for z/OS was cataloged without selecting the Enable Encryption option under the Server Authentication part of the Security Options section. This is the default behavior when you catalog a DB2 database in DB2 Configuration Assistant. Figure 8-17 shows this option in the Configuration Assistant panel.

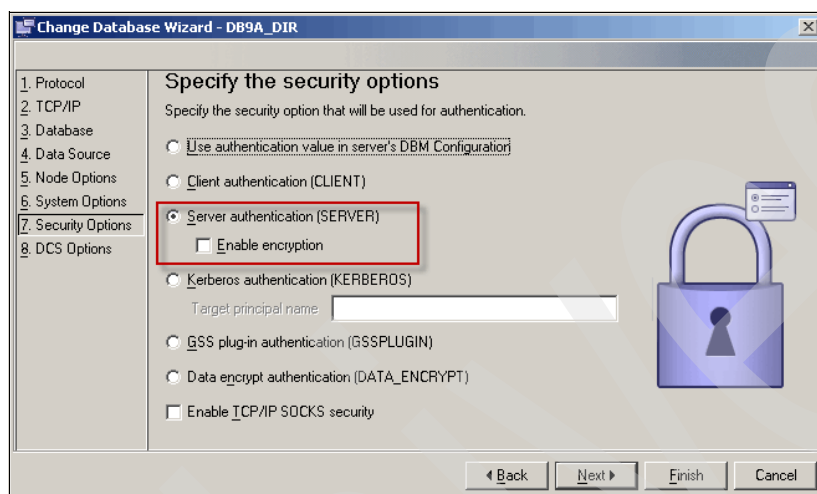


Figure 8-17 Configuration Assistant server authentication: No encryption

The Network analyzer was started during some tests that included a connection between the workstations and DB2 for z/OS. Using the Wireshark interface, we browsed the DRDA packets and we expanded the properties. We can see that the DRDA DDM code point

SECMEC (security mechanism) is x'0003', which is *User ID and Password Security Mechanism*. However, as shown in Figure 8-18, the password is moved across the network in clear text.

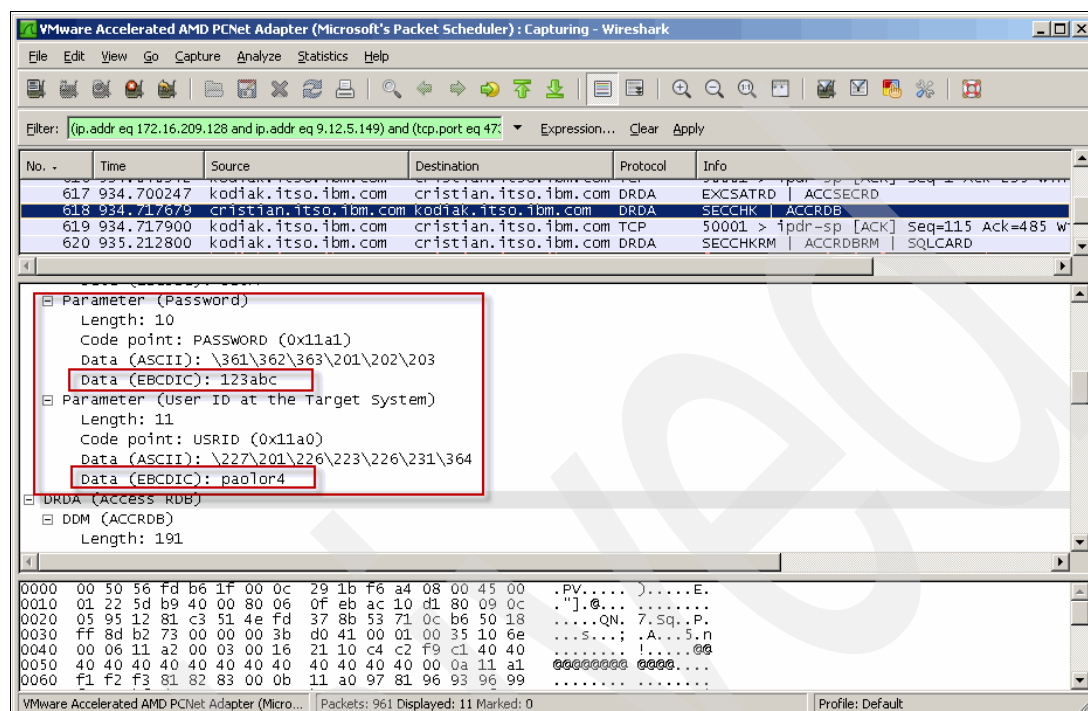


Figure 8-18 Password and user ID in network analyzer

Example 8-46 shows a text report of this DRDA communication, and you can see that the user *paolor4* is using the password *123abc*.

#### Example 8-46 DRDA SECCHK Non encrypted example

```
...
Parameter (Relational Database Name)
  Length: 22
  Code point: RDBNAM (0x2110)
  Data (ASCII): \304\302\371\301@@@@@@@@@@@@@@
  Data (EBCDIC): DB9A
Parameter (Password)
  Length: 10
  Code point: PASSWORD (0x11a1)
  Data (ASCII): \361\362\363\201\202\203
  Data (EBCDIC): 123abc
Parameter (User ID at the Target System)
  Length: 11
  Code point: USRID (0x11a0)
  Data (ASCII): \227\201\226\223\226\231\364
  Data (EBCDIC): paolor4
....
```

Figure 8-19 Configuration assistant enable encryption option

This situation is usually not acceptable. You need to consider the Enable Encryption option shown in Figure 8-20.

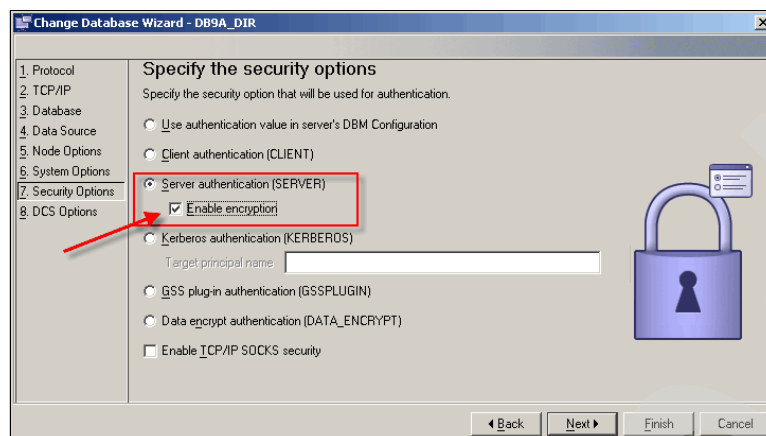


Figure 8-20 Configuration Assistant Server Authentication, Enable encryption option selected

After changing the security options, the same connection shows that SECMEC = x'0009' in the DRDA trace indicating *Encrypted User ID and Password Security Mechanism* according to the DRDA DDM definitions. Example 8-47 shows an extract of the text version of the trace. Password and user ID are not visible in the network trace.

Example 8-47 DRDA SECCHK Encrypted enabled example

```
....
Parameter (Relational Database Name)
  Length: 22
  Code point: RDBNAM (0x2110)
  Data (ASCII): \304\302\371\301@@@@@@@@@@@@@@
  Data (EBCDIC): DB9A
Parameter (Security Token)
  Length: 12
  Code point: SECTKN (0x11dc)
  Data (ASCII): \333\264K\224\004]\251\226
  Data (EBCDIC): ...m\004)zo
Parameter (Security Token)
  Length: 12
  Code point: SECTKN (0x11dc)
  Data (ASCII): /\226G\
  Data (EBCDIC): /\+o.*
....
```

## 8.4 DB2 for z/OS tracing capabilities

The following traces are commonly used to monitor and gather performance data about a DB2 subsystem and its applications.

- ▶ Statistics trace
- ▶ Accounting trace
- ▶ Monitor trace
- ▶ Performance trace
- ▶ Global trace

We discuss them at high level hereafter in the context of a distributed database environment. For details about the topics in this section, refer to the following publications:

- ▶ *DB2 Version 9.1 for z/OS, Performance Monitoring and Tuning Guide*, SC18-9851
- ▶ *DB2 Version 9.1 for z/OS, Command Reference*, SC18-9844

For information about IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, refer to the following IBM publications:

- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS and IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS Report Reference Version 4.1*, SC18-9984-01
- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS and IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS Report Command Reference*, SC18-9985-02
- ▶ *A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS*, SG24-7224.

This book provides details that will help you install and understand the main functions of the product, clarify the differences, and point out the advantages if you still have one of the pre-existing products already in use.

## 8.4.1 Collecting traces

The DB2 command `START TRACE` starts DB2 traces. This command can be issued from a z/OS console, a DSN session, a DB2I panel, an IMS or CICS terminal or a program using the Instrumentation Facility Interface (IFI).

Example 8-48 shows part of this command syntax. Refer to the DB2 documentation for details on this command and its options.

*Example 8-48 START TRACE command syntax, partial*

---

```
>>-START TRACE--(+--PERFM---+--)------>
      +-ACCTG---+
      +-STAT----+
      +-AUDIT---+
      '-MONITOR-'

      .-,-----|.
      V          |
>--DEST--(---+GTF-+---)-+-----+-----+-----+----->
      +-SMF-+      '-constraint block-' '-RMID-'
      +-SRV-+
      +-OPn-+
      '-OPX-'

>--+-----+-----+-----+-----><
  '-COMMENT(string)-' |           .-LOCAL-. |
                      '-SCOPE-(+--GROUP-+--)-'
```

---



Example 8-49 shows how to start a PERFORMANCE trace with destination System Management Facility (SMF). In this case, we are interested on collecting IFCID 180 which contains the DRDA communication buffers. You may also use GTF as trace destination.

*Example 8-49 Starting IFCID 180*

---

```
-STA TRA(PERF) DEST(SMF) CLASS(30) IFCID(180)
```

---

Example 8-50 shows the output of this command. Take note of the number assigned by DB2 to the trace (in this case 03), because you will need it to modify the trace later on.

*Example 8-50 START TRACE output example*

---

```
DSNW130I  -DB9A PERFORMANCE TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSN9022I  -DB9A DSNWVCM1 '-STA TRA' NORMAL COMPLETIO
**
```

---

Example 8-51 shows the output of the DISPLAY TRACE command. The command DISPLAY TRACE lists all the active traces. Note the TNO 03, this is the trace started in Example 8-49. Use this command to verify the destination and the IFCID.

*Example 8-51 DIS TRACE command example*

---

```
DSNW127I  -DB9A CURRENT TRACE ACTIVITY IS -
TNO TYPE  CLASS      DEST QUAL IFCID
01  STAT   01,03,04,05, SMF  NO
01              06
02  ACCTG   01,02,03,07, SMF  NO
02              08
03  PERF 30              SMF  NO   180
*****END OF DISPLAY TRACE SUMMARY DATA*****
DSN9022I  -DB9A DSNWVCM1 '-DIS TRA' NORMAL COMPLETION
```

---

After executing the workload on which you want the trace collection, stop the trace as indicated in Example 8-52. You need to identify the trace type and the trace number, TNO(3).

*Example 8-52 STOP TRACE command example*

---

```
-STO TRACE(PERF) TNO(3)
```

---

If you are using SMF as the destination, you need to consider that the SMF data sets contain a lot more information than just DB2. It typically contains records related to all other subsystems and applications. Information stored in SMF data sets are critical to any IT shop and these data sets must be handled carefully. SMF data can be stored for several months or years depending on the system and applications requirements. Refer to your local system programmers and administrators for SMF data management.

Example 8-53 on page 366 shows how you can use the switch SMF command to manually start the dump of the contents of a currently active SMF data set. The recording of new SMF data is transferred from one SMF data set to another. All SMF data in storage is written out before the transfer is made.

You may need to do this when you require to analyze the SMF data immediately and do not want to wait until completion of the next normal SMF dump cycle, which would take place automatically when the active SMF data set becomes full.

### Example 8-53 Switch SMF command

```
/I SMF
```

Example 8-54 shows the output of the SMF dump process. The SMF dump program issues the message IFA020I once for each input and output data set. In this example the ddname DUMPALL, dsname SMFDATA.ALLRECS.G0007V00, contains the SMF records listed in the Summary Activity Report section.

### Example 8-54 Identify SMF dump data set from SMF Dump job

```
IFA010I SMF DUMP PARAMETERS
IFA010I END(2400) -- DEFAULT
IFA010I START(0000) -- DEFAULT
IFA010I DATE(1900000,2099366) -- DEFAULT
IFA010I OUTDD(DUMPALL,TYPE(0:255)) -- SYSIN
IFA010I INDD(DUMPIN,OPTIONS(ALL)) -- SYSIN
IFA020I DUMPALL -- SMFDATA.ALLRECS.G0007V00
IFA020I DUMPIN -- SYS1.SC63.MAN1
IFA018I SMF DATA SET DUMPIN HAS BEEN CLEARED.
```

SUMMARY ACTIVITY REPORT							
START DATE-TIME 04/01/2009-18:15:45				END DATE-TIME 04/02/2009-15:21:10			
RECORD TYPE	RECORDS READ	PERCENT OF TOTAL	AVG. RECORD LENGTH	MIN. RECORD LENGTH	MAX. RECORD LENGTH	RECORDS WRITTEN	
2	0					1	
3	0					1	
4	44	.10 %	215.00	215	215	44	
5	42	.09 %	147.09	145	153	42	
14	959	2.14 %	443.08	344	804	959	
15	426	.95 %	344.64	344	380	426	
17	93	.21 %	100.00	100	100	93	
18	64	.14 %	144.00	144	144	64	

Example 8-55 shows how you can extract the DB2 SMF records from a dump SMF data set. This example creates a file, ddname DUMPOUT, which contains only DB2 SMF records 100, 101, 102. This helps in reducing resource consumption during reporting. Additional parameter can be added in the ddname SYSIN to further filter the records to be copied to the DUMPOUT ddname.

### Example 8-55 Extract DB2 SMF records from SMF dump dataset

```
//PA0L044B JOB (999,POK),REGION=5M,MSGCLASS=X,CLASS=A,
//          MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
/*JOBPARM S=SC63
/*-----
/* DRDA REDBOOK --> EXTRACT SMF RECORDS
/*-----
//STEP1 EXEC PGM=IFASMFDP
/* IFASMFDP UTILITY COPIES THE INPUT SMF DATA TO OUTPUT DATASETS
//DUMPIN DD DISP=SHR,DSN=SMFDATA.ALLRECS.G0007V00
//DUMPOUT DD DISP=(NEW,CATLG),DSN=PA0LOR4.SMF.DUMP01,
//          SPACE=(CYL,(500,40,),RLSE),
//          DCB=(RECFM=VBS,BLKSIZE=4096,LRECL=32760),
//          UNIT=3390
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
          INDD(DUMPIN,OPTIONS(DUMP))
          OUTDD(DUMPOUT,TYPE(100:102))
/*
```

## Using GTF as a trace destination

The default destination for the performance trace classes is the Generalized Trace Facility (GTF), but it can be used as the destination for other traces.

Using GTF as trace destination allows for separation from standard SMF use and provides results that can be used immediately for analysis, while using SMF as destination usually requires to wait till next SMF dump data set creation. The following steps show how to start and stop the collection of traces using GTF as destination.

You need to create a procedure to start a GTF trace collector. Example 8-56 shows the procedure we used.

*Example 8-56 SYS1.PROCLIB(GTFDRDA), a GTF proclib example*

---

```
//GTFDRDA PROC MEMBER=GTFDRDA
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
// TIME=1440,REGION=2880K
//IEFRDER DD DSN=PAOLOR4.GTFDRDA.&SYSNAME..D&YYMMDD..T&HHMMSS,
// DISP=(,CATLG),SPACE=(CYL,666,RLSE),UNIT=SYSDA,
// VOL=SER=SB0X49
//SYSLIB DD DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

---

For the space allocation of the target data set, ddname IEFRDER in Example 8-56, consider that GTF only allocates primary extents, not secondary extents. Example 8-57 shows the parmlib member that we created for this example.

*Example 8-57 SYS1.PARMLIB(GTFDRDA), GTF option example*

---

```
TRACE=USRP
USR=(FB9)
```

---

We start the GTF collector by executing the command S GTFDRDA.GTFDRDA, as shown in Example 8-58.

*Example 8-58 Start GTF system log output*

---

```
IEA989I SLIP TRAP ID=X33E MATCHED. JOBNAME=*UNAVAIL, ASID=0091.
S GTFDRDA.GTFDRDA
$HASP100 GTFDRDA ON STCINRDR
IEF695I START GTFDRDA WITH JOBNAME GTFDRDA IS ASSIGNED TO USER STC
, GROUP SYS1
$HASP373 GTFDRDA STARTED
IEF403I GTFDRDA - STARTED - TIME=18.40.15 - ASID=0091 - SC63
AHL121I TRACE OPTION INPUT INDICATED FROM MEMBER GTFDRDA OF PDS
SYS1.PARMLIB
TRACE=USRP
USR=(FB9)
AHL103I TRACE OPTIONS SELECTED --USR=(FB9)
*054 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
```

---

You record DB2 trace data in GTF using a GTF event ID of X'FB9'.

Note message AHL125A. During the initialization, GTF issues this message to allow you to accept or reject the trace option.

Message AHL080I then lists the GTF storage used. See Example 8-59 on page 368.

*Example 8-59 Start GTF: Answer to message AHL125A*

---

```
R 54, U
IEE600I REPLY TO 054 IS; U
U
AHL080I GTF STORAGE USED FOR GTF DATA: 151
      GTFBLOCK STORAGE      82K BYTES (BLOK=      40K)
      PRIVATE STORAGE      1038K BYTES (SIZE=     1024K)
      SADMP HISTORY        54K BYTES (SADMP=      40K)
      SDUMP HISTORY        54K BYTES (SDUMP=      40K)
      ABEND DUMP DATA      0K BYTES (ABDUMP=      0K)
AHL031I GTF INITIALIZATION COMPLETE
```

---

Example 8-60 shows the command for starting the trace of the DRDA related IFCIDs 180,183 and 184 with destination GTF. Refer to Table 8-5 on page 372 for default values and destinations for traces. Note that we used CLASS(30) to avoid starting other IFCIDs.

*Example 8-60 START TRACE with destination GTF*

---

```
-STA TRA(PERF) DEST(GTF) CLASS(30) IFCID(180,183,184)
COMMENT('DRDA TRACE PROB INVEST')

DSNW130I -DB9A PERF TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSN9022I -DB9A DSNWVCM1 '-STA TRA' NORMAL COMPLETION
***
```

---

After the start of the trace, you need to reproduce the problem. Once recorded, It is recommended to stop the trace before stopping GTF. Example 8-61 shows the interactions with the console required for stopping the GTF traces using during this test. When you stop the GTF, you must specify the additional identifier used at startup, GTFDRDA in our example.

*Example 8-61 STOP GTF trace*

---

```
STOP GTFDRDA
AHL006I GTF ACKNOWLEDGES STOP COMMAND
AHL904I THE FOLLOWING TRACE DATASETS CONTAIN TRACE DATA : 155
      PAOL0R4.GTFDRDA.SC63.D090413.T224015
-                                     --TIMINGS (MINS.)--
      ----PAGING COUNTS---
-JOBNAME  STEPNAME  PROCSTEP  RC  EXCP   CPU   SRB  CLOCK  SERV
PG  PAGE  SWAP    VIO  SWAPS  STEPNO
-GTFDRDA  GTFDRDA  IEFPROC    00   38   .00   .00   7.12   588
 0      0      0      0      0      1
IEF404I GTFDRDA - ENDED - TIME=18.47.23 - ASID=0091 - SC63
-GTFDRDA ENDED. NAME-                TOTAL CPU TIME=   .00
TOTAL ELAPSED TIME=  7.12
$HASP395 GTFDRDA  ENDED
```

---

When the STOP command takes effect, the system issues message AHL006I. If the system does not issue message AHL006I, then GTF tracing continues.

When the GTF records are written to disk, the GTF writes to a wrap-around data set; if the destination file allocation is too small for the amount of data, data will be overlapped.

If no wrap-around occurred, you should find the START TRACE and STOP TRACE commands as the second and previous to last record in the generated file.

## Performance overhead

Traces have an associated performance overhead and you can expect the following trace costs:

- ▶ Statistics traces

Negligible, because the statistics trace record is written only at the statistics time interval specified in minutes as a DB2 systems parameter. The default value for this DSNZPARM STATIME is 5. It is recommended to use STATIME=1.

- ▶ Accounting traces

Accounting traces typically incur the following overhead:

- Class 1: Typically, less than 5% transaction overhead, except for fetch-intensive applications that do not use multi-row fetch
- Class 2: CPU overhead can be 20% for fetch-intensive applications
- Class 3: Less than 1% CPU overhead
- Class 7 and 8: Less than 5% CPU overhead

- ▶ Audit Traces

Audit traces typically incur less than 10% transaction overhead with all classes on.

- ▶ Performance traces class 1, 2, and 3

Performance might incur the following overhead:

- 5% to 100% CPU overhead
- Performance class 3 CPU overhead can be as high 100% for fetch-intensive applications

- ▶ Global Trace

Global trace can incur 10 to 150% CPU overhead

Some traces can introduce a huge impact on performance, but you can limit the impact by tracing only selected threads by specifying, for instance, a PLAN or AUTHID with the START TRACE command. This is known as using a constraint block. Consider to take advantage of this technique to control the performance impact of starting traces.

The constraint block places optional constraints on the kinds of data that are collected by the trace. The allowable constraints depend on the type of trace started, as shown in Table 8-4.

Table 8-4 Allowable constraints for each trace type

Type	PLAN	AUTHID	CLASS	LOCATION
PERFM	Allowed	Allowed	Allowed	Allowed
ACCTG	Allowed	Allowed	Allowed	Allowed
STAT	NO	NO	Allowed	NO
AUDIT	Allowed	Allowed	Allowed	Allowed
MONITOR	Allowed	Allowed	Allowed	Allowed

Furthermore, it is possible to use excluding filters by using a filtering directive to the start trace command. For instance the XPLAN(WAS001) option will trace all plans but will exclude the plan WAS001. DB2 9 for z/OS has added many filtering options, refer to *DB2 Version 9.1 for z/OS Command Reference*, SC18-9844 for a detailed list of the available trace qualifiers.

The LOCATION option of the START TRACE command can be used for specifying a list of specific location names for which trace information is gathered. DB2 for z/OS does not receive a location name from clients that are not DB2 for z/OS subsystems. To start a trace for a client that is not a DB2 for z/OS subsystem, enter its LUNAME or IP address. Enclose the LUNAME by the less-than (<) and greater-than (>) symbols. Enter the IP address in the form nnn.nnn.nnn.nnn.

An example of starting some specific IFCIDs for a DB2 Connect Server Edition which is connecting to z/OS using the IP address 9.12.5.149 is shown in Example 8-62.

*Example 8-62 Use of LOCATION for starting tracing for non z/OS clients*

---

```
-STA TRA(PERF) DEST(SMF) CLASS(30) IFCID(180,183,184)
LOCATION(9.12.5.149) COMMENT('DRDA TRACE KODIAK GTW')

DSNW130I  -DB9A PERF TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSN9022I  -DB9A DSNWVCM1 '-STA TRA' NORMAL COMPLETION
***
```

---

Consider using a constraint block with the START TRACE command to control the impact of starting traces.

When starting a trace, you can use the IFCID option to specify which other IFCIDs, in addition to those IFCIDs contained in the classes specified in the CLASS option, are to be started. To start only those IFCIDs specified in the IFCID option, use trace classes 30–32. These classes have no predefined IFCIDs and are available for a location to use. If you do not specify the IFCID option, only those IFCIDs contained in the activated trace classes are started.

**Attention:** To minimize performance impacts, use CLASS 30, 31 or 32 with the IFCID option when starting a single IFCID instead of starting the CLASS that contains this IFCID. This is particularly important when working with PERFORMANCE traces.

## Minimizing the effects of traces on DB2 performance

The volume of data DB2 trace collects can be quite large. Consequently, the number of trace records that you request affect system performance. When activating a performance trace, you should qualify the START TRACE command with the particular classes, plans, authorization IDs, and IFCIDs you want to trace.

General considerations to minimize the effects of traces on DB2 performance are as follows:

- ▶ When starting a performance trace, be sure that you know what you want to report. I/O only or SQL only, for example. See OMEGAMON PE documentation for examples of which classes produce which reports. Otherwise, you might have incomplete reports and have to rerun the measurements or collect too much data, overloading the data collector.
- ▶ Use the default statistics frequency of five minutes. When the statistics trace is active, statistics are collected by SMF at all times.
- ▶ Decide if the continuous collection of accounting data is needed. If a transaction manager provides enough accounting information, DB2 accounting might not be needed. In environments where the processor is heavily loaded, consider not running accounting on a continuous basis.
- ▶ When using accounting on a continuous basis, start classes 1 and 3 to SMF (SMF ACCOUNTING on installation panel DSNTIPN). You might also want to start accounting class 2 because it provides additional information that can be useful in resolving problems. Accounting class 2 does introduce some additional processor cost.

- ▶ To capture package accounting information with minimal impact, start accounting trace classes 7 and 8. If you need more detailed accounting information for packages, such as SQL statements and locking, you can temporarily add accounting trace class 10.
- ▶ Package accounting introduces additional CPU cost and increases the amount of SMF accounting trace records.
- ▶ If you need only minimal accounting information for packages, start account accounting.
- ▶ Use the performance trace for short periods of time (START/STOP TRACE) and restrict it to certain users, applications, and classes. Use the default destination GTF to allow immediate analysis of the trace information.
- ▶ Start the global trace only if a problem is under investigation, and IBM service personnel have requested a trace.

## Working with traces

DB2 for z/OS provides the facilities for collecting and recording DB2 data and events through the Instrumentation Facility Component (IFC), it does not provide, however, tools for analysis and reporting. These tasks must take place outside DB2.

After collecting the traces, you can analyze and report them by using the following tools:

- ▶ Tivoli OMEGAMON XE for DB2  
You can use it to format, print, and interpret DB2 trace output. You can view an online snapshot from trace records by using OMEGAMON or other online monitors. Refer to the publication *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, Report Reference*, SC18-9984 and *Report Command Reference*, SC18-9985 for details
- ▶ OMEGAMON Performance Warehouse  
It provides an infrastructure at the OMEGAMON Server and at the workstation to automate performance analysis tasks. It introduces the concept of processes which represent single or recurring tasks such as loading DB2 data into the Performance Warehouse or generating reports. The definition of processes and analysis tasks can be performed at the workstation through the Performance Warehouse graphical user interface.
- ▶ Custom made parsing tools  
If you want to do your own analysis of the trace output, or you do not have access to a trace analysis and reporting product, you can use the information described in *DB2 Version 9.1 for z/OS, Performance Monitoring and Tuning Guide*, SC18-9851 and the trace field descriptions that are shipped with DB2. By examining a DB2 trace record, you can determine the type of trace that produced the record (statistics, accounting, audit, performance, monitor, or global) and the event the record reports.

Each trace class captures information about several subsystem events. These events are identified by many instrumentation facility component identifiers (IFCIDs). The IFCIDs are described by the comments in their mapping macros, contained in hlq.SDSNMACS and hlq.DSNWMSGs, which is shipped with DB2.

The DESTINATION option of the START TRACE command specifies where the trace output is to be recorded. You can use more than one value, but do not use the same value twice. If you do not specify a value, the trace output is sent to the default destination shown in Table 8-5 on page 372. This table shows the most common destination for each trace type.

If the specified destination is not active or becomes inactive after you issue the START TRACE command, you receive message DSNW133I, which indicates that the trace data is lost.

Table 8-5 Most common allowable destinations for each trace type

Type	GTF	SMF	OPX
PERFM	Default	Allowed	Allowed
ACCTG	Allowed	Default	Allowed
STAT	Allowed	Default	Allowed
AUDIT	Allowed	Default	Allowed
MONITOR	Allowed	Allowed	Default

It is important to understand the impact on the system and on your analysis work of the selected destination of traces. Refer to the publication *DB2 Version 9.1 for z/OS, Performance Monitoring and Tuning Guide*, SC18-9851 for details.

## 8.4.2 Using OMEGAMON PE for reporting

We used OMEGAMON PE for the reports shown on this book. For information about Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, refer to the following Web page:

<http://www.ibm.com/software/tivoli/products/omegamon-xe-db2-peex-zos/>

The online documentation can be found at the following Web page:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe\\_db2.doc/ko2welcome\\_pe.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm)

For instructions about getting and installing the client, refer to the following Web page:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe\\_db2.doc/ko2welcome\\_pe.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm)

The following Web page lists the tables used in the OMEGAMON Performance Warehouse:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe\\_db2.doc/ko2ccb1202.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2ccb1202.htm)

For details about how you can use OMEGAMON XE for performance and statistics analysis, refer to *A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS*, SG24-7224.

Example 8-63 shows part of Accounting Trace report —short.

Example 8-63 OMEGAMON PE Accounting Trace report—short

1 LOCATION: DB9A		OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)						PAGE: 1-37	
GROUP: N/P		ACCOUNTING TRACE - SHORT						REQUESTED FROM: ALL 00	
MEMBER: N/P								TO: DATES 23	
SUBSYSTEM: DB9A								ACTUAL FROM: 03/30/09 15	
DB2 VERSION: V9								PAGE DATE: 03/31/09	
PRMAUTH	CORRNMBR	ACCT	TIMESTAMP	SELECTS	DELETES	MERGES	CPU TIME(CL1)	GETPAGES	TOT.PREF
PLANNAME	CONNECT	TERM.	CONDITION	OPENS	INSERTS	PREPARE	EL. TIME(CL2)	BUF.UPDT	LOCK SUS
CORRNAME	THR.TYPE		COMMITTS	FETCHES	UPDATES	EL. TIME(CL1)	CPU TIME(CL2)	SYN.READ	LOCKOUTS
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
PAOLOR4	exe	18:57:23.827983		0	0	0	0.038651	119	7
TotoTest	SERVER	NORM TYP2 INACT		1	0	0	0.134929	12	0



Example 8-64 shows part of an OMEGAMON PE Accounting Trace report—long.

**Example 8-64** OMEGAMON PE Accounting Trace report—long

```

LOCATION: DB9A                      OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)
GROUP: N/P                        ACCOUNTING TRACE - LONG
MEMBER: N/P
SUBSYSTEM: DB9A
DB2 VERSION: V9

PAGE: 1-243
REQUESTED FROM: ALL 00
TO: DATES 23
ACTUAL FROM: 03/30/09 15

----- IDENTIFICATION -----
ACCT TSTAMP: 03/31/09 18:57:25.68  PLANNAME: TotoTest      WLM SCL: DDFDEF      CICS NET: N/A
BEGIN TIME : 03/31/09 18:57:23.91  PROD TYP: COMMON SERV  CICS LUN: N/A
END TIME   : 03/31/09 18:57:25.68  PROD VER: V9 R5 M0     CICS INS: N/A
REQUESTER  : ::9.30.28.151          CORRNAME: msqry32.     LUW NET: G90C0646
MAINPACK   : SYSSTAT               LUW LUN: J03B          LUW INS: C3F8267F270C
PRIMAUTH   : PAOLOR4               CONNTYPE: DRDA         LUW SEQ: 2
ORIGAUTH   : PAOLOR4               CONNECT : SERVER      ENDUSER : Toto
                                           TRANSACT: TotoTestApplication
                                           WSNAME  : TotoMac

----- DISTRIBUTED ACTIVITY -----
REQUESTER : ::9.30.28.151  ROLLBCK(1) RECEIVED: 0  PREPARE RECEIVED : 0
PRODUCT ID : COMMON SERV  SQL RECEIVED : 10  LAST AGENT RECV. : 0
PRODUCT VERSION : V9 R5 M0 COMMIT(2) RESP.SENT: 0  THREADS INDOUBT : 0
METHOD : DRDA PROTOCOL  BACKOUT(2)RESP.SENT: 0  MESSAGES IN BUFFER : 5910

.....

----- INITIAL DB2 COMMON SERVER OR UNIVERSAL JDBC DRIVER CORRELATION -----
PRODUCT ID : COMMON SERV
PRODUCT VERSION: V9 R5 M0
CLIENT PLATFORM: NT
CLIENT APPLNAME: TotoTestApplication
CLIENT AUTHID : Toto
DDCS ACC.SUFFIX: TotoAccountingInfo

```

## OMEGAMON PE Performance Database

The Performance Database is a DB2 database that can hold aggregated DB2 trace information spanning a long period of time. The performance data can come from the following data groups:

- ▶ Accounting
- ▶ Audit
- ▶ Locking
- ▶ Record traces
- ▶ Statistics and system parameters
- ▶ Batch, periodic, and display exceptions.

To help you build the Performance Database, OMEGAMON PE provides the sample library RKO2SAMP with Data Definition Language (DDL) definitions called C-parts. Additional metadata files are provided that describe the table columns (called B-parts). These files are in a format that allows their content to be loaded into metadata tables for querying the table layouts.

Performance data to be loaded into the Performance Database and Performance Warehouse must be prepared by the reporting functions using the report language subcommands FILE and SAVE. The FILE subcommand generates sequential data sets that can be loaded to a database directly. The SAVE subcommand creates VSAM data sets that must be converted with save-file utilities to sequential data sets that are in loadable format. The sample library RKO2SAMP provides metadata files that describe the record layouts of the sequential data sets (so-called D-parts). These files are in a format that allows their contents to be loaded into metadata tables for querying the record layouts. DB2 load statements are available in

RKO2SAMP that can be used with a LOAD utility to store the sequential data set contents into the database tables (called L-parts).

Using the Performance Database and Performance Warehouse, you can analyze the health of your DB2 subsystems and applications and perform trend analysis or capacity planning based on historical data. You can write your own SQL queries and analysis applications, or you can use queries and rules-of-thumb provided in OMEGAMON PE.

The Performance Warehouse provides a set of SQL queries and rules-of-thumb (ROTs) that help you to identify complex performance problems. Besides the standard queries and ROTs that come with the product, you may also define and use your own ROTs and write your own SQL queries to analyze the performance data.

Figure 8-21 depicts the process used for populating the OMEGAMON warehouse tables under demand and creating custom spreadsheet-based reports.

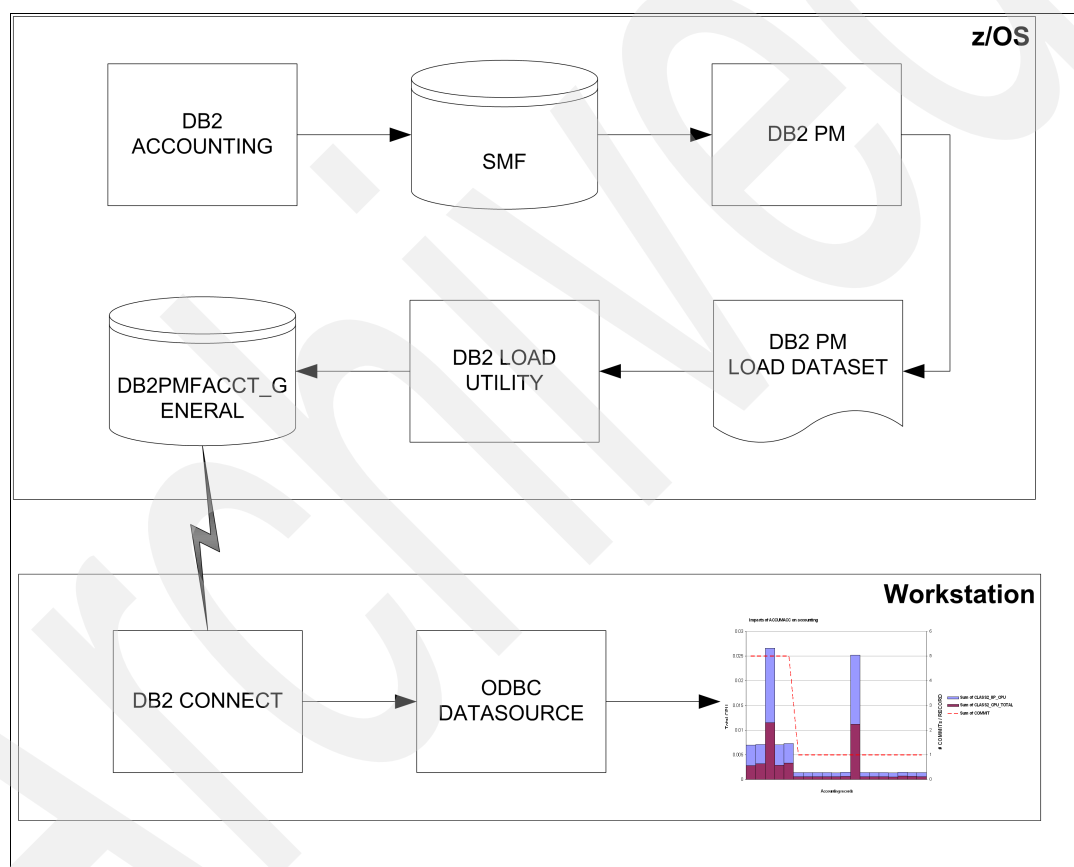


Figure 8-21 Creating spreadsheets reports from OMEGAMON Warehouse tables

The steps involved on this methods are as follows:

1. After collecting traces on SMF or GTF, execute an OMEGAMON batch report including the creation of a file that can be loaded in the warehouse tables
2. Execute the LOAD utility using the Load Control Statement that maps the file with the warehouse tables. The required load control statements are provided by OMEGAMON PE and are mentioned in the following lines.
3. Once the tables loaded, access the data using ODBC from a workstation. The report that you can create is limited by your imagination and the graphical capabilities of the reporting tool. One powerful option is the exploitation of Excel Pivot Tables.

The following tables contains information that can help you in the process of problem determination.

- ▶ **DB2PMFACCT\_GENERAL**: DB2 table to store DB2PM Accounting FILE data category GENERAL
  - DDL is located in hlq.RKO2SAMP(DGOACFGE)
  - File record layout description located in hlq.RKO2SAMP(DGOABFGE)
  - DB2 Load utility control statements located in hlq.RKO2SAMP(DGOALFGE)
- ▶ **DB2PMFACCT\_DDF**: DB2 table to store DB2PM Accounting FILE data category DDF
  - DDL is located in hlq.RKO2SAMP(DGOACFDF)
  - File record layout description located in hlq.RKO2SAMP(DGOADFDF)
  - DB2 Load utility control statements located in hlq.RKO2SAMP(DGOALFDF)
- ▶ **DB2PM\_STAT\_DDF**: DB2 table to store DDF Statistics
  - DDL is located in hlq.RKO2SAMP(DGOSCDDF)
  - File record layout description located in hlq.RKO2SAMP(DGOSBDDF)
  - DB2 Load utility control statements located in hlq.RKO2SAMP(DGOSLDDF)

Figure 8-22 shows an example of custom report based on the table DB2PM\_STAT\_DDF. It shows the number of block sent per DB2 subsystem per statistics interval (5 minutes in our example.) We compare D9C1 (DB2 not using blocks) with DB9A (DB2 using blocks).

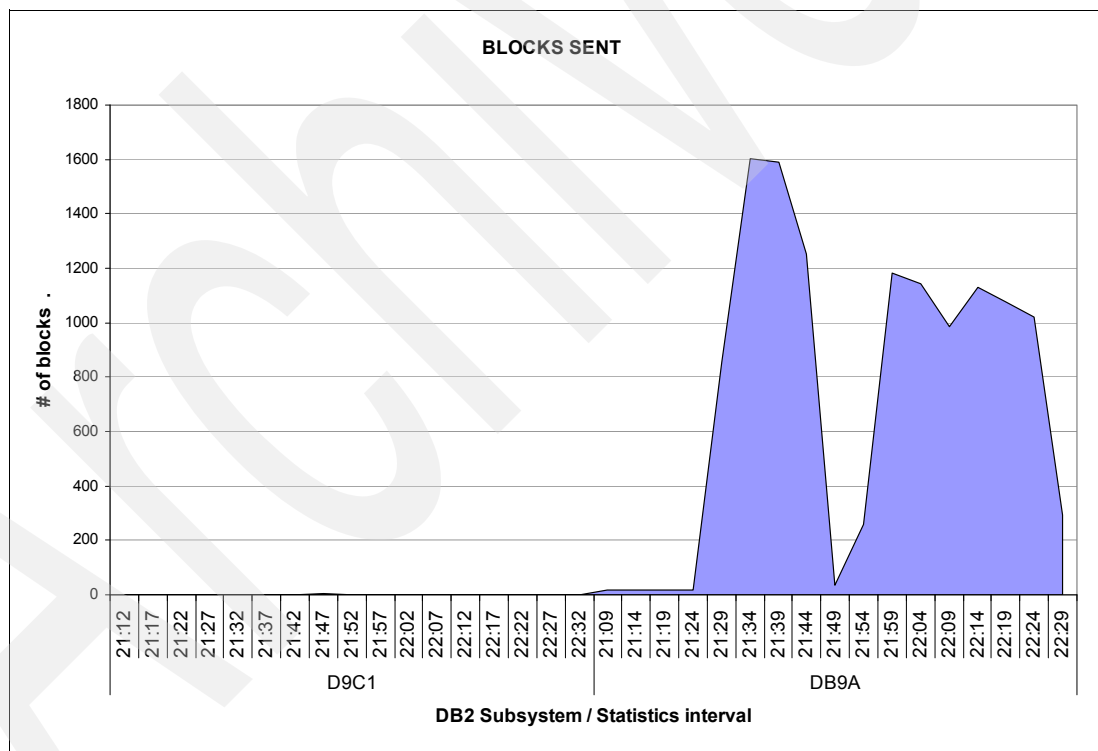


Figure 8-22 Custom DDF Statistics chart, blocks sent by statistics interval

Figure 8-23 shows an example of a chart created based on the table DB2PMFACCT\_DDF. This example shows the DBAT waiting time versus the number of commits executed during a test scenario per location. A high value in DBAT WAIT TIME may suggest the need for increasing the value of the DSNZPARM MAXDBAT.

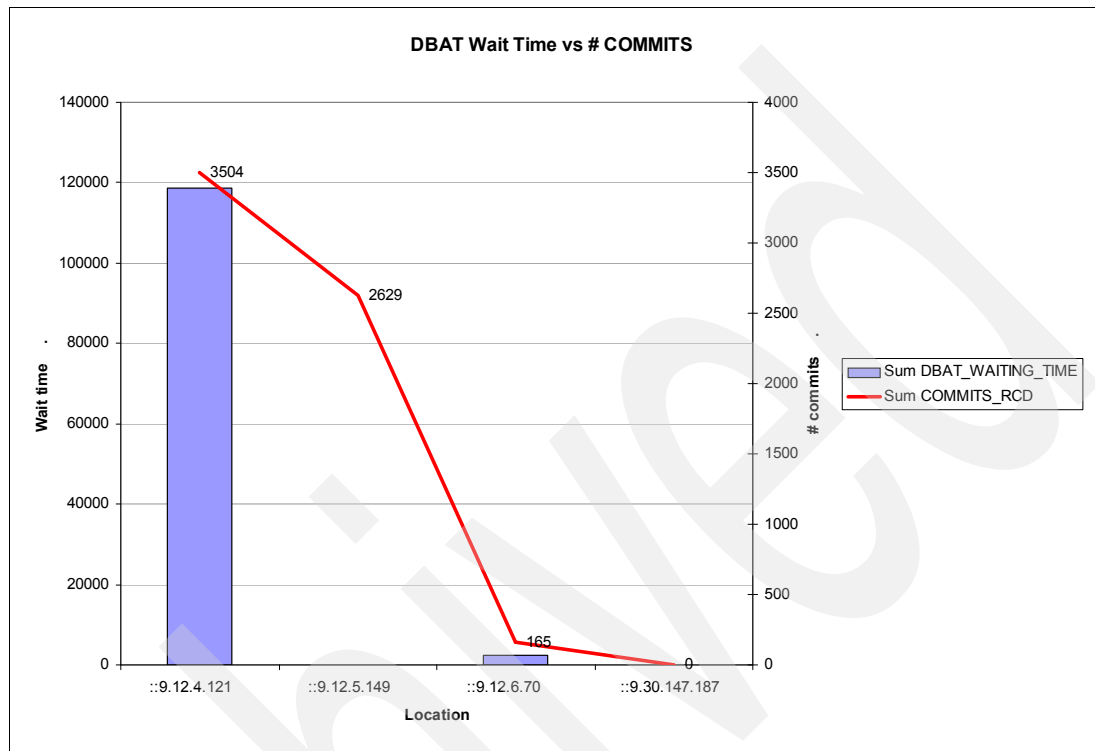


Figure 8-23 Custom DDF accounting chart, DBAT wait time versus commits per location

Figure 8-24 is an example of report based on the table DB2PMFACCT\_GENERAL. It shows the zIIP utilization by Client Workstation Name for a given period of time

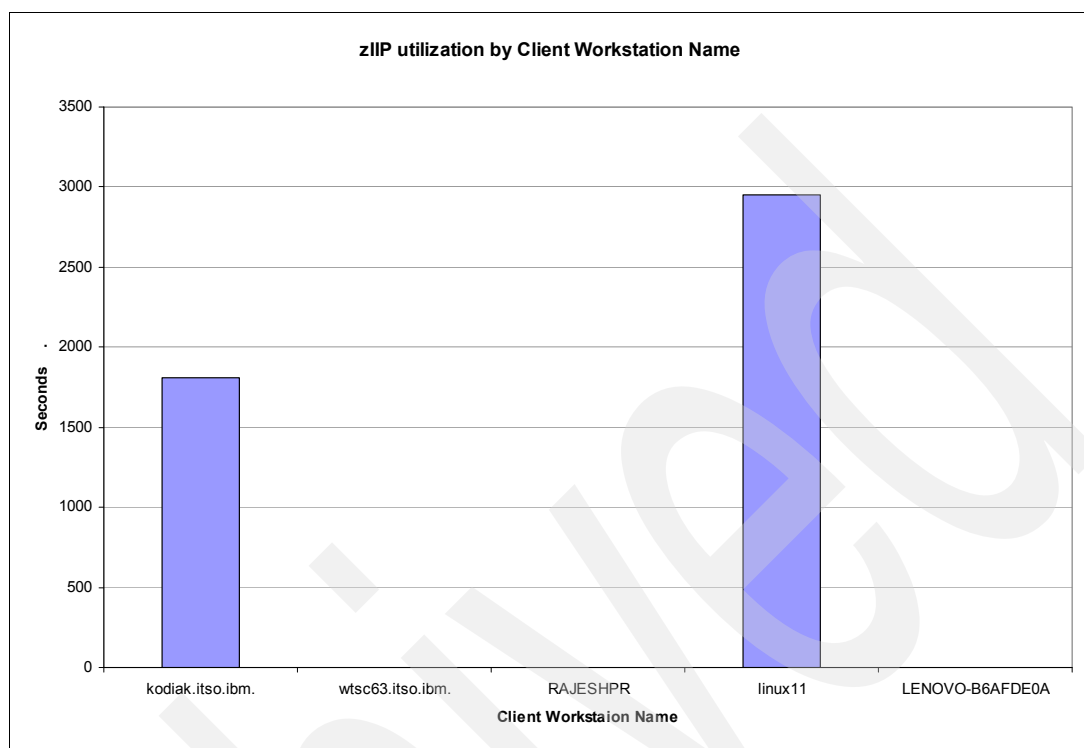


Figure 8-24 Analysis of zIIP utilization by workstation name

### 8.4.3 IFCIDs for DRDA problem determination

When a DB2 trace is active, internal events trigger the creation of trace records. The records, identified by Instrumentation Facility Component Identifiers (IFCIDs), can be written to buffers, or to file destinations like SMF and GTF records. IFCIDs written to buffers are in general exploited by online monitors that may start traces when you access a panel that requires so. In general you use commercial batch reports or custom programs for creating reports from IFCIDs stored on files.

The following list include the most important IFCIDs for analyzing distributed workloads.

- IFCID 2, 3. Statistics and Accounting

Not specific for distributed workload, new counters were added through the APAR PK62161 to show number of rows involved in multi-row operations. DB2 for z/OS V8 and DB2 9 for z/OS users were unable to measure the gains due to performance enhancements such as multi-row fetch and insert because the relevant counters were not part of the DB2 statistics (IFCID 2) and DB2 accounting (IFCID 3) records. The new counters added to the DB2 statistics record provide the per system values that correspond to the number of rows fetched, inserted, updated, and deleted. The counters added to the DB2 accounting records provide the same values per user.

OMEGAMON PE supports PK62161 with PK89128 (PTF UK47981) as illustrated in Example 8-65 on page 378.

The SQL CALL DATA section will contain the following fields:

- NUMBER OF ROWS FETCHED
- NUMBER OF ROWS INSERTED
- NUMBER OF ROWS UPDATED
- NUMBER OF ROWS DELETED

**Example 8-65 OMEGAMON PE report example: Rows involved in multi rows operations**

1	LOCATION: DB9A			OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4R2)					PAGE: 1-17	
	GROUP: N/P			RECORD TRACE - LONG					REQUESTED FROM: NOT SPECIFIED	
	MEMBER: N/P								TO: NOT SPECIFIED	
	SUBSYSTEM: DB9A								ACTUAL FROM: 05/01/09 17:33:57.70	
	DB2 VERSION: V9								PAGE DATE: 05/01/09	
OPRMAUTH	CONNECT	INSTANCE	END_USER	WS_NAME			TRANSACTION			
ORIGAUTH	CORRNAME	CONNTYPE	RECORD TIME	DESTNO ACE	IFC	DESCRIPTION	DATA			
PLANNAME	CORRNMBR		TCB CPU TIME	ID						
-----										
N/P	N/P	C41DCABDDFF5	N/P	N/P			N/P			
N/P	N/P	'BLANK'	17:34:10.49447284	10251	2	2 DB STATISTICS	NETWORKID: DB9A	LUNAME: DB9A	LUWSEQ: 1	
N/P	N/P	N/P	N/P							
-----										
SQL CALL DATA										
SELECT	5	LOCK TABLE	0	CREATE TABLE	1	ALTER TABLE	0	DROP TABLE	1	
INSERT	204	DESCR.TABLE	0	CREATE INDEX	0	ALTER INDEX	0	DROP INDEX	0	
UPDATE	3334	SET RULES	0	CREATE TSPAC	0	ALTER TSPAC	0	DROP TSPAC	0	
DELETE	0	SET HOST VAR	4	CREATE DBASE	0	ALTER DBASE	0	DROP DBASE	0	
PREPARE	9710	SET CURR.SQL	1808	CREATE STGRP	0	ALTER STGRP	0	DROP STGRP	0	
OPEN	5639	SET CUR.PREC	0	CREATE SEQ	0	ALTER SEQ	0	DROP SEQ	0	
CLOSE	5631	SET CURR.DEG	0	CREATE FUNC	0	ALTER FUNC	0	DROP FUNC	0	
FETCH	14787	SET PATH	0	CREATE PROC	1	ALTER PROC	0	DROP PROC	1	
DESCRIBE	2465	SET CONNECT	0	CREATE ALIAS	0			DROP ALIAS	0	
GRANT	0	CONNECT TYP1	0	CREATE DIST	0			DROP DIST	0	
REVOKE	0	CONNECT TYP2	1	CREATE SYNON	0			DROP SYNON	0	
RELEASE	0	ASOC LOCATOR	0	CREATE VIEW	0	ALTER VIEW	0	DROP VIEW	0	
COMMENT ON	0	HOLD LOCATOR	0	CREATE TRIG	0	INCRMT BIND	1	DROP TRIG	0	
LABEL ON	0	FREE LOCATOR	0	CREATE T.TAB	0	DECL T.TAB	0	DROP PACKAGE	0	
		ALLOC CURSOR	0	CREATE A.TAB	0	RENAM TABLE	0			
MERGE	0	RENAME IX	0	CREATE ROLE	0	ALTER JAR	0	DROP ROLE	0	
TRUNC TABLE	0		0	CREATE TRUST	0	ALTER TRUST	0	DROP TRUST	0	
NUMBER OF ROWS FETCHED				21124						
NUMBER OF ROWS INSERTED				10033						
NUMBER OF ROWS UPDATED				3344						
NUMBER OF ROWS DELETED				19						

Example 8-66 shows the PL/I program we used for creating the multi-row insert workload used for the analysis of the trace records.

**Example 8-66 Sample PL/I program: multi row insert**

```
//jobcard
//*-----
/* MULTIPLE ROW INSERT
//*-----
//PC EXEC DSNHPLI, MEM=INSERT5, USER=PAOLOR4,
// PARM.PC='HOST(PLI), XREF, STDSQL(NO), CONNECT(2) '
//PPLI.SYSIN DD *
INSERT5: PROC OPTIONS (MAIN);
EXEC SQL INCLUDE SQLCA;
EXEC SQL INCLUDE SQLDA;

/* DECLARATIONS */

DCL MYSTR1 CHAR(900) VARYING;
DCL MYSTR2 CHAR(100) VARYING;
DCL MYSTR3 CHAR(900) VARYING;

DCL XUSER CHAR(8) INIT('PAOLOR4');
DCL XPASS CHAR(8) INIT('123ABC');
```

```

DCL XDB2 CHAR(4) INIT('DB9C');

DCL HCHAR1(5) CHAR(3);
DCL HCHAR2(5) CHAR(1);
DCL HCHAR3(5) CHAR(10);
DCL HIND BIN FIXED(15);
DCL INT1 BIN FIXED(15);
DCL CNTR BIN FIXED(15) INIT(0);

HCHAR1(1) = 'ABC';
HCHAR1(2) = 'ABC';
HCHAR1(3) = 'ABC';
HCHAR1(4) = 'ABC';
HCHAR1(5) = 'ABC';
HCHAR2(1) = 'A';
HCHAR2(2) = 'A';
HCHAR2(3) = 'A';
HCHAR2(4) = 'A';
HCHAR2(5) = 'A';
HCHAR3(1) = 'ABCDEFGHIJ';
HCHAR3(2) = 'ABCDEFGHIJ';
HCHAR3(3) = 'ABCDEFGHIJ';
HCHAR3(4) = 'ABCDEFGHIJ';
HCHAR3(5) = 'ABCDEFGHIJ';
INT1 = 5;

/* CONNECT TO TARGET DB2 FOR Z/OS DB2 */

EXEC SQL CONNECT TO DB9A USER :XUSER USING :XPASS ;
CALL PUT_SQLCA('CONNECTION',0);

MYSTR2 = 'FOR MULTIPLE ROWS';
MYSTR1 = 'INSERT INTO MYTABLE VALUES(?,?,?)';

EXEC SQL PREPARE STMT1 ATTRIBUTES :MYSTR2 FROM :MYSTR1;
CALL PUT_SQLCA('INSERT5: PREPARE STMT',0);

DO WHILE (CNTR < 200);

EXEC SQL EXECUTE STMT1 USING :HCHAR1, :HCHAR2, :HCHAR3 FOR
:INT1 ROWS;

CALL PUT_SQLCA('INSERT5: EXECUTE STMT',0);

CNTR = CNTR + 1;

END;

/* FORMAT THE SQLCA FIELDS OF INTEREST AND PUT TO FILE SYSPRINT. */

PUT_SQLCA: PROC(AT_STMT,EXPECTED_CODE);
DCL AT_STMT CHAR(30), EXPECTED_CODE FIXED BINARY(15);
PUT SKIP EDIT('STMT:',AT_STMT,
' EXPECTED SQLCODE:',EXPECTED_CODE,
' ACTUAL SQLCODE:',SQLCA.SQLCODE)

```

```

(A,A,A,A,A,A);
END;

END /* MAIN PROCEDURE BLOCK */;
/*
//BINDLOC EXEC PGM=IKJEFT01
//DBRMLIB DD DISP=SHR,DSN=PAOLOR4.DBRMLIB.DATA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(D9CG)
BIND PACKAGE(PKGSAMP) VALIDATE(RUN) ACTION(REP) -
MEMBER(INSERT5) RELEASE(COMMIT)
BIND PACKAGE(DB9A.PKGSAMP) VALIDATE(RUN) ACTION(REP) -
MEMBER(INSERT5) RELEASE(COMMIT)
BIND PLAN(PLNSAMP) ACTION(REP) -
PKLIST(*.PKGSAMP.INSERT5) VALIDATE(RUN) -
RELEASE(COMMIT)
END
/*
//RUN EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(D9CG)
RUN PROGRAM(INSERT5) PLAN(PLNSAMP) LIBRARY('PAOLOR4.RUNLIB.LOAD')
END

```

---

► IFCID 53, 58, 59

These are not distributed workload specific IFCIDs. They can be used for block fetch investigation. Prior to DB2 V8, one IFCID 59 was out for every row fetched, starting with DB2 V8, 1 IFCID 59 is created for every block fetched.

► IFCID 191, 192, 193, 195

Distributed specific IFCIDs, often report DRDA exceptions and are associated with a DSNLxxxx console message.

► IFCID 203, 204, 205, 235, 238

Distributed specific IFCIDs relating to RESTART.

Refer to the IBM documentation for details on these IFCIDs and to the OMEGAMON PE documentation for a description of the related reports.

You can use the OMEGAMON PE RECTRACE command to generate record traces and file data sets. Example 8-67 shows the syntax for a OMEGAMON PE trace report including IFCID 192.

*Example 8-67 OMEGAMON PE trace report example, IFCID(192)*

```

//SYSIN      DD *
    RECTRACE
        TRACE LEVEL(LONG) DDNAME(RTTRCDD1)
        INCLUDE( IFCID(192))
    DB2PM EXEC
/*

```

---



Example 8-68 shows a portion of the report generated with this command.

**Example 8-68 OMEGAMON PE IFCID 192 report example**

LOCATION: DB9A  
GROUP: N/P  
MEMBER: N/P  
SUBSYSTEM: DB9A  
DB2 VERSION: V9

PRIMAUTH CONNECT  
ORIGAUTH CORRNAME  
PLANNAME CORRNMBR

INSTANCE  
CONNTYPE

END\_USER  
RECORD TIME  
TCB CPU TIME

WS\_NAME  
DESTNO ACE IFC  
ID

DESCRIPTION  
DATA

TRANSACT  
DATA

SYSOPR DB9A C3F91E62A25F 'BLANK' 'BLANK' 'BLANK'

SYSOPR 028.DBA 'BLANK' 13:26:24.93597053 3910 1 192 DDM LEVEL 6A NETWORKID: G90C0646 LUNAME: J03B LUNSEQ: 1

'BLANK' 02 N/P REQUESTING LOCATION: ::9.30.28.156 REQUESTING TIMESTAMP: N/P AR NAME: 'BLANK' PRDID: N/P

REMOTE LOCATION: ::9.30.28.156  
ERROR TYPE: SYNTAX  
CURRENT 6A HEADER  
PREVIOUS 6A HEADER  
FIRST 250

SEVERITY: X'00000008'  
OFFSET  
X'5FFE6F50'  
X'00000000'

VERSION NUMBER: 1  
GDS LENGTH  
0  
0

CSECT: DSNLIRTR  
DDM CONST  
X'00'  
X'00'

FLAG  
X'00'  
X'00'

REQ CORR  
0  
0

0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

0020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

0040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

0060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

0080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00E0 00000000 00000000 00000000 00000000 00000000 00000000 0000 0000

## Creating a trace reporting tool

The following steps help you create a custom IFCID reporting tool. For this example, we used GTF traces as input to a REXX program that was created with the purpose of reporting the contents of the IFCID 180. The IFCID 180 records the DRDA Communication Buffers for versions prior to DB2 9, this record is for IBM service.

The following section describes the steps needed to analyze and report the DRDA communication buffer messages using the IFCID 180 and a GTF trace as input. The REXX code of this program can be found in D.2, "REXX parser of GTF trace for IFCID 180" on page 446.

OMEGAMON PE currently (V4.2) supports reporting of IFCID but the communication buffers are not parsed. You cannot see the DRDA messages in the report. A future version of OMEGAMON PE may parse the DRDA messages.

**Important:** This is an example of mapping the IFCID 180 provided for your convenience. This tool was tested for the purpose of this book and the validation performed was limited. Use this program as a training tool and at your own risk.

### The GTF header section

The macro hlq.SDSNMACS(DSNDQWGT) describes the GTF header. An extract is shown in Example 8-69. This macro contains definitions for PLS and assembler, but this information can be easily adapted to be used in a REXX program.

**Example 8-69 GTF Trace header extract of hlq.SDSNMACS(DSNDQWGT)**

QWGT HDR	DS	OF	/*MVS GTF HEADER	*/
QWGT LEN	DS	H	/*LENGTH OF THE RECORD	*/
	DS	H	/*RESERVED	*/
QWGT AID	DS	X	/*APPLICATION IDENTIFIER -FF	*/

```

QWGTFFID DS    X          /*FORMAT ID - EITHER F5 OR ZERO          */
QWGTTIME DS    XL8 /*TIMESTAMP-TIME=YES MUST BE STATED ON GTF START */
QWGTEID  DS    XL2       /*EVENT ID - 0FB9              */
QWGTASCB DS    A         /*ASCB ADDRESS              */
QWGTJOBN DS    CL8       /*JOB NAME                  */
*
* /* ....THE FOLLOWING INFORMATION IS USED TO SPAN RECORDS...      */
QWGTHDRE DS    OF        /*EXTENTION TO THE HEADER   */
QWGTDLEN DS    H         /*LENGTH OF THE DATA SECTION */
QWGTDZZ  DS    OXL2      /*CONTROL BYTES              */
QWGTDSCC DS    X         /*SEGMENT CONTROL CODE       */
*
QWGTD00 EQU    0         /*COMPLETE RECORD           */
QWGTD01 EQU    1         /*FIRST SEGMENT              */
QWGTD02 EQU    2         /*LAST SEGMENT               */
QWGTD03 EQU    3         /*MIDDLE SEGMENT             */
*
QWGTDZZ2 DS    X         /*RESERVED                   */
QWGTSID  DS    CL4       /*SUBSYSTEM ID               */
QWGTWSEQ DS    F         /*SEQUENCE NUMBER - THE SAME NUMBER IS USED FOR ALL
*                          RECORDS ASSOCIATED WITH A SPANNED SET OF RECORDS.
*                          QWHSWSEQ IS EQUAL TO QWGTWSEQ.          */
QWGTEND  DS    OF        /* END OF WRITER SECTION     */
*

```

**Tip:** If you are writing a REXX program for parsing GTF or SMF information, you can concentrate on the offset of the fields of interest in the definition macros.

As indicated in the macro, you must take the following considerations into account:

- ▶ GTF must be started with TIME=YES. The mapping macro assumes the time stamp is in the GTF-supplied header
- ▶ If the DB2 supplied data is greater than 256 bytes the records are spanned by DB2. The installation is responsible for reblocking the records before the mapping macros are used.

Table 8-6 contains an extract of the Summary of Constants table that can be found in the IBM publication *z/Architecture Reference Summary*, SA22-7871. This information can help you to understand the data in the GTF record as described by the mapping macros.

Table 8-6 Extract of constants definitions, *z/Architecture® Reference Summary*

Type	Implied length, bytes	Default alignment	Format
A	4	Word	Value of address
H	2	Halfword	Fixed-point binary
F	4	Word	Fixed-point binary
X	-	Byte	Hexadecimal digits

GTF records are blocked to 256 bytes. Because some of the trace records exceed the GTF limit of 256 bytes, they are blocked by DB2. The following steps are described in the IBM publication *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

1. Is the GTF event ID of the record equal to the DB2 ID (that is, QWGTEDID = X'x'FB9')?
  - If it is not equal, get another record.
  - If it is equal, continue processing.

- If it is spanned (that is, QWGTDS00 <> QWGTDS01), test to determine whether it is the first, middle, or last segment of the spanned record.
  - If it is the first segment (that is, QWGTDS00 = QWGTDS01), save the entire record including the sequence number (QWGTWSEQ) and the subsystem ID (QWGTSSID)
  - If it is a middle segment (that is, QWGTDS00 = QWGTDS02), find the first segment matching the sequence number (QWGTWSEQ) and on the subsystem ID (QWGTSSID). Then move the data portion immediately after the GTF header to the end of the previous segment
  - If it is the last segment (that is, QWGTDS00 = QWGTDS03), find the first segment matching the sequence number (QWGTWSEQ) and on the subsystem ID (QWGTSSID). Then move the data portion immediately after the GTF header to the end of the previous record

Process the completed record.

- If it is not spanned, process the record.

Example 8-70 shows 3 lines of the GTF trace file used for the current example. This example show the data in HEX Mode on, With HEX Mode on, all characters are displayed in hexadecimal format. The editor replaces each actual character with its two-byte hexadecimal equivalent. You can change any character by changing its two-byte code.

```

=COLS>  ---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8---+---9---+---0---+---1---
000003  D i  ?00% 9m PAOLOR4 i DB9A  ROLE ( * )  ê j A!\DB9AD i  Â  DB9A  USIBM
OC08431EDEB0F90DCDDDDF40A00CCFC0001DDDC4454520000000000050100091C5ECFC00843161000000010000CCFC444444444444EECCD
0489095A6F90940716369400A204291000796350DCD0000045800002100421F1A0429148909526000700070004429100000000000042924

-----
000004  D i . '  60% 9m PAOLOR4 S DB9A  @ b  â I  ÌDB9C  DSNESM68  ✕
OC084731CEB0F90DCDDDDF40E00CCFC00010007080000030400C007CCFC444444444444444444444444444444444400CEDCEDFF0000000000090
0489BD71EF9094071636940020042910008000C020100080201900842930000000000000000000000000000000004255246831000000001F0

-----
000005  D i . ' Î% 0% 9m PAOLOR4  DB9A  i  Q S  n ^  ^  ¶  > ; PAOLOR4  .BATCH
OC0847760EB0F90DCDDDDF40000CCFC000100010A0000030D00E400000000000000FF0000009B0B000B100B140615DCDDDDF4444444CECC4
0489BD5CECF909407163694010104291000900100A01000808012000030000000000FF9C694B500E000C0006010E1E716369400000B213380

-----

```

Figure 8-25 shows an example of mapping the GTF header of an IFCID record. As exposed by the field QWGTDSCC, the example records (IFCID 180) are spanned in two records. They need to be assembled before parsing the data section containing the actual IFCID information.

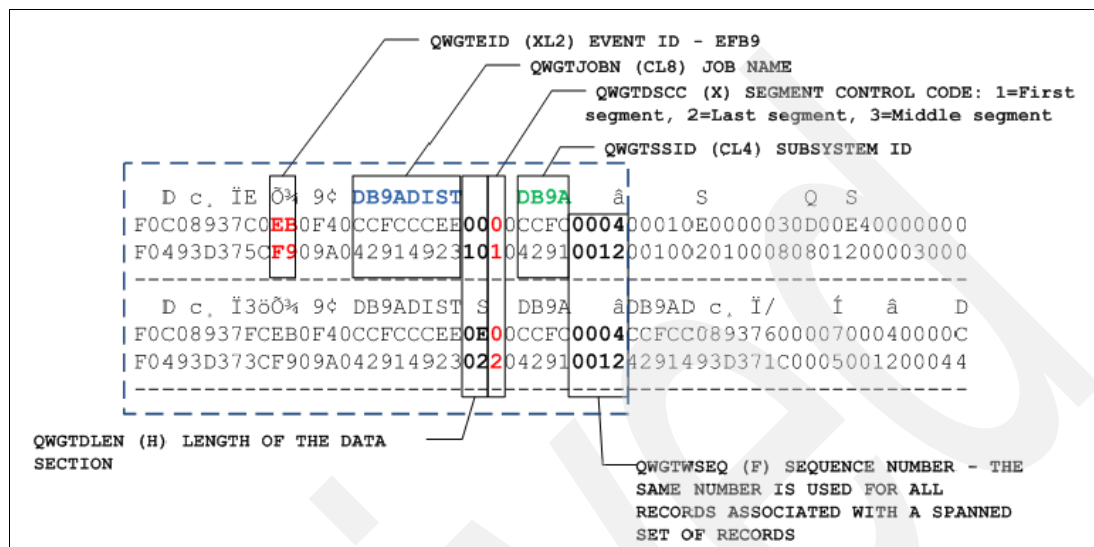


Figure 8-25 Mapping the GTF header

The self-defining section, which follows the writer header, contains pointers that enable you to find the product and data sections, which contain the actual trace data. Following the method described above and the information shown in Example 8-71, we were able to produce the reporting section of the example REXX program. Example 8-71 describes the fields of the IFCID 180.

**Example 8-71 GTF trace header extract of hlq.SDSNMACS(DSNDQW02), IFCID 180**

```
*****
*   IFCID 0180 FOR RMID 27 RECORDS - DC COMMUNICATION BUFFERS   *
*   *                                                             *
*   DC WILL TRACE THE VTAM BUFFER LIST ENTRIES THAT ARE SENT OR  *
*   RECEIVED.  EACH BUFFER LIST ENTRY WILL TRACE NO MORE THAN 4096 *
*   BYTES OF DATA. IFCID 0180 SHOULD BE ROUGHLY IDENTICAL TO IFCID 0184 *
*****
*
QW0180  DSECT          IFCID(QWHS0180)
*
QW0180HD DS    OF      HEADER - ALL TYPES
QW0180E  DS    CL1     TYPE OF EVENT - A, R, S, or F
*
*   **.....QW0180E  CONSTANTS.....**
QW0180ER EQU    C'R'   RECEIVED A DISTRIBUTED DATA MESSAGE
QW0180ES EQU    C'S'   SENT A DISTRIBUTED DATA MESSAGE
QW0180EF EQU    C'F'   RECEIVED AN FMH-5 ON AN INCOMING CONV.
QW0180EA EQU    C'A'   SENT AN FMH-5 TO ALLOCATE A CONVERSATION
*
QW0180F  DS    XL1     FLAGS
QW0180NP EQU    X'CO'  NETWORK PROTOCOL.
*                          APPLICABLE ONLY IF QWHSRN>=91
*                          '00'b - SNA (S).
```

*		'01'b - TCP/IP (T) IPv4.
*		'10'b - TCP/IP (T) IPv6.
QW0180CV DS	OCL20	CONVERSATION INFORMATION
QW0180VI DS	XL4	(S) CONVERSATION ID
*		(T) SOCKET DESCRIPTOR
QW0180IP DS	XL16	(T) IF QWHSRN>=91, IP ADDRESS
*		(INTERNAL FORM). If QW0180NP='01'b
*		THEN IT CONTAINS AN IPv4 ADDRESS
*		WHICH MAY BE MAPPED.
*		IF QW0180NP='10'b, then
*		IT CONTAINS A 128 BIT IPv6 ADDRESS.
	ORG QW0180IP	
QW0180SI DS	XL8	(S) SESSION ID
*		(T) If QWHSRN < 91, 32 BIT IPv4
*		ADDRESS, FOLLOWED BY 16 BIT LOCAL
*		PORT NUMBER, FOLLOWED BY 16 BIT
*		PARTNER PORT NUMBER.
QW0180LM DS	CL8	(S) MODE NAME
QW0180PT DS	XL4	(T) IF QWHSRN >= 91, PORTS
*		(INTERNAL FORM).
ORG QW0180PT		
QW0180LP DS	XL2	(T) LOCAL PORT
QW0180PP DS	XL2	(T) PARTNER PORT
QW0180DL DS	H	SIZE OF QW0180DS VARIABLE LENGTH DATA
QW0180DS DS	OC	VARIABLE LENGTH MESSAGE OR FMH-5 DATA
*		(PASSWORD IN FMH-5 IS CHANGED TO BLANKS)
	SPACE 2	
*		

The key in the parsing of the DRDA communication buffers is to understand how the DDM objects are defined. All DDM architecture can be broken down into small, well-defined, and standardized units called objects. Objects are data processing entities. Example of DDM objects are password, userid, commit, prepare SQL statement, and so forth.

All objects are composed of a contiguous string of bytes with a specific format that describes the object. This format is composed as follows:

- ▶ Length
- ▶ Codepoint
- ▶ Data

The length is a 2-byte binary value that contains the length of the object. The length value is always the first part of any object. The value of the length field includes the following information:

- ▶ Length of the length field
- ▶ Length of the codepoint field
- ▶ Total length of the object's data

The codepoint is a 2-byte hexadecimal value that indicates where the class description of the object can be found in a dictionary. The data area of an object is composed of one or more contiguous bytes. These bytes may contain the following information:

- ▶ Scalars byte strings over which one or more data values have been mapped.
- ▶ Collections objects that contain one or more objects in their data area.

Refer to the Open Group publication *DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture* for more information.

Example 8-72 shows the JCL used for calling this REXX program. The ddname TRACES contains the GTF traces. This program will report unknown information and will continue processing. For instance, in DB2 for z/OS 9 and earlier IFCID 180 trace records can also contain Private Protocol flows, these records will be ignored by the tool. This REXX supports the following parameters:

- ▶ DB2 subsystem (DB9A in our example)
- ▶ Trace type identifier (only GTF is supported)
- ▶ DATA.

This parameter indicates if the actual buffer data of the codepoint is to be printed in binary, EBCDIC and ASCII format. DATA=Y will print the data, DATA=N will not.

*Example 8-72 JCL sample: calling IFCID 180 formatting tool*

---

```

/*-----
/* DRDA REDBOOK --> REXX FOR FORMATING GTF TRACES
/*-----
//RUN      EXEC PGM=IKJEFT01
//SYSPROC DD  DISP=SHR,DSN=PAOLOR4.DRDA.UTIL.CNTL
//SYSPRINT DD  SYSOUT=*
//SYSTSPT DD  SYSOUT=*
//TRACES   DD  DISP=SHR,DSN=PAOLOR4.GTFDRDA.XXX
/* REXX NAME
/*
/*      DB2 SUBSYSTEM
/*      |      TRACE TYPE
/*      |      |      DISPLAY DATA OPTION (DATA=Y | DATA=N)
/*      V      V      V      V
/* PARSESEX DB9A  GTF DATA=Y
//SYSTSIN DD  *
PARSEREX DB9A GTF DATA=N
/*

```

---

Example 8-73 shows a portion of the output when the option DATA=N is selected.

*Example 8-73 IFCID 180 formatting example, DATA=N option*

---

```

(REC) =====> EXCSAT - Exchange Server Attributes
(REC) =====> EXTNAM - External Name
(REC) =====> MGRLVLLS - Manager-level List
(REC) =====> SRVCLSNM - Server Class Name
(REC) =====> SRVNAM - Server Name
(REC) =====> SRVRLSLV - Server Product Release Level
(REC) =====> ACCSEC - Access Security
(REC) =====> SECMEC - Security Mechanism
(REC) =====> RDBNAM - Relational Database Name
<===== (SND) EXCSATRD - Server Attributes Reply Data
<===== (SND) EXTNAM - External Name
<===== (SND) MGRLVLLS - Manager-level List
<===== (SND) SRVCLSNM - Server Class Name
<===== (SND) SRVNAM - Server Name
<===== (SND) SRVRLSLV - Server Product Release Level
<===== (SND) ACCSECRD - Access Security Reply Data
<===== (SND) SECMEC - Security Mechanism

```

```

(REC) =====> SECCHK - Security Check
(REC) ----->  SECMEC - Security Mechanism
(REC) ----->  RDBNAM - Relational Database Name
...

```

---

Example 8-74 shows the output when DATA=Y is selected.

*Example 8-74 IFCID 180 formatting example, DATA=Y option*

---

```

(REC) =====> EXCSAT - Exchange Server Attributes
(REC) ----->  EXTNAM - External Name

+-- H E X -----+ +-- EBCDIC -----+ +-- ASCII -----+
|84 82 F2 82 97 4B 85 A7| 85 40 40 40 40 40 40 40| |db2bp.exe| | | |
|40 40 40 40 F0 F2 F6 F0| F0 F2 F3 F8 F0 F0 F0 00| | 02600238000| | | |
|00 00 00 00 00 00 00 00| 00 00 00 00 00 00 00 00| | | | |
|00 00 00 00 00 00 00 00| 00 00 00 00 00 00 00 60| | | | |
|F0 F0 F0 F1 D4 D6 D3 C1| D9 D6 40 40 40 40 40 40| |0001MOLARO| |(|< |
|40 40 40 40 40 40 40 40| 40 40 40 40 40 40 40 40| | | | |
|40 40 C4 C2 F9 C1 6D C4| C9 D9 F0 C4 C2 F2 40 40| |DB9A_DIR0DB2| | | |
|40 40 40 40 40 40 40 40| 40 40 40 00 18 .. ..| |...| |...|
+-----+ +-----+ +-----+

(REC) -----> MGRLVLLS - Manager-level List

+-- H E X -----+ +-- EBCDIC -----+ +-- ASCII -----+
|14 03 00 07 24 07 00 09| 14 74 00 05 24 0F 00 08| | | | |
|14 40 00 08 00 0B .. ..| .. .. .. .. ..| | .....| |"d" H d p|
+-----+ +-----+ +-----+

(REC) -----> SRVCLSNM - Server Class Name

+-- H E X -----+ +-- EBCDIC -----+ +-- ASCII -----+
|D8 C4 C2 F2 61 D5 E3 00| 0C .. .. .. ..| |QDB2/NT .....| | + .....|
+-----+ +-----+ +-----+

```

---

## 8.4.4 WLM classification rules and accounting information

z/OS is able to manage many type of workloads, each one with different business importance and processing characteristics. To accomplish this, the installation must categorize the incoming work to the system using the classification rules.

Classification rules are the filters that WLM uses to associate a transaction's external properties (also called work qualifiers, such as LU name or user ID) with a goal.

Figure 8-26 on page 388 represents how WLM classifies incoming workload into a service class (or a reporting class) following the definitions of the classification rules. For DB2-specific performance information for WLM, see *DB2 Version 9.1 for z/OS Administration Guide*, SC18-9840.

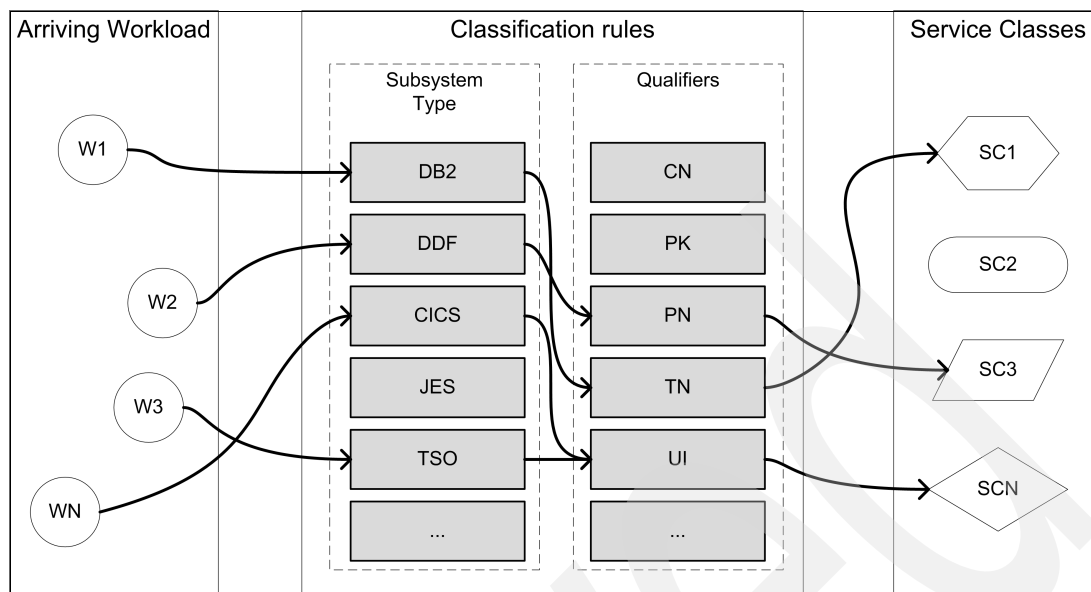


Figure 8-26 WLM workload classification

Table 8-7 shows an extract of the qualifiers that can be used in the WLM classification rules. Not all work qualifiers are valid for every subsystem type. They are subsystem dependent. For details on which qualifiers are valid for which subsystems, and a full list of qualifiers, see the *z/OS V1R1.0 MVS Planning: Workload Management*, SA22-7602-00.

Table 8-7 Extract of work qualifiers and their abbreviations

Qualifier	Details
AI	Accounting information
CI	Correlation information
CN	Collection name
PK	Package name
PN	Plan name
PR	Procedure name
SI	Subsystem instance
SY	System name
TN	Transaction name/job name
UI	Userid

DDF receives requests from different clients: DB2 and other vendors' software, each transaction becomes an enclave. An enclave is an independent dispatchable UOW, which is basically a business transaction that can span multiple address spaces, and can include multiple SRBs and TCBs. DB2 uses enclaves for work coming into the system through DDF. (DB2 also uses enclaves for other purposes, like CPU parallelism.) The enclave is created by DDF for an incoming connection when the first SQL statement starts to execute.



To understand best practices for DDF workload classification, it is important to know the DB2 DDF thread and enclave relationship. Database access threads have two modes of processing:

- **ACTIVE MODE**

A database access thread is always active from initial creation to termination.

- **INACTIVE MODE**

A database access thread can be active or pooled. (A database access thread that is not currently processing a UOW is called a pooled thread, and it is disconnected.) When a database access thread in INACTIVE MODE is active, it processes requests from client connections within UOWs. When a database access thread is pooled, it waits for the next request from a client to start a new UOW.

The CMTSTAT subsystem parameter (DDF THREADS installation field) specifies whether to make a thread active or inactive after it successfully commits or rolls back and holds no cursors.

Because WLM assigns the performance goals to the enclaves, it is the lifetime of the enclave that WLM takes as the duration of the work. Therefore, when you run with CMTSTAT=INACTIVE, DDF creates one enclave per transaction, and response time goals and multiple time periods can be used. However, if you have CMTSTAT=ACTIVE or CMSTAT=INACTIVE, DDF creates one enclave for the life of the thread.

You need to classify each request. If you do not classify your DDF transactions into service classes, they are assigned to the default class SYSOTHER, which has a discretionary goal. It is normally a good idea to have different goals for mission-critical DDF work (higher goals), and batch-like DDF activity (lower goals).

**Important:** If you do not define any classification rule for DDF requests, all enclaves get the default service class SYSOTHER. This is a default service class normally reserved to low priority work

It is highly recommended to use INACTIVE MODE to take advantage of WLM's ability to manage each individual UOW according to its business goals.

For a detailed description and samples of how to classify DB2 Stored Procedures, refer to *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083.

## **Considerations for JDBC requests**

JDBC sends its requests to DB2 using Driver Type 2 or Driver Type 4.

The work of Type 2 comes from WebSphere Application Server to DB2 and it is classified under Component Broker (CB) subsystem type. You are not required to classify it under DDF subsystem type.

For the Type 4/DDF workload, the DDF rules allow us to do the classification. Not all of the DDF qualifiers are usable. For example, all the JDBC applications use the same packages so it is not a useful filter. If you want to base prioritization on application behavior, you can classify based upon the first stored procedure called in a transaction, but it is only usable if you call a stored procedure and the call of the stored procedure is the first call to DB2 (when the enclave needs to be created and the work classified).

A simple approach is to use a different data source within WebSphere Application Server for each classification and programmatically use a specific data source. Each data source has

either a different AUTHID associated with it or a JDBC driver collection. You can classify based upon either of those choices.

The only other approach is to use the DB2 client strings. The client strings are text attributes associated with the connection, which we can use for classification of the workload. The client strings can be set as a part of the data source definition or alternatively, you can set them programmatically within the application so that every transaction can have a different value. We can use the client strings for workload classification, and we can also use them for end-to-end auditing. (The fields are written in DB2 accounting reports)

### 8.4.5 Step-by-step performance analysis

Before you attempt to fine-tune applications, you need to understand the application and system architecture, as well as the customer business environment. There can be no single tool that can substitute the knowledge a person holds while troubleshooting problems. Tools help to identify bottlenecks in an n-tier system, and some tools provide possible recommendations to identify the bottlenecks further and aid in the resolving them.

Figure 8-27 depicts our approach to problem determination in an n-tier environment.

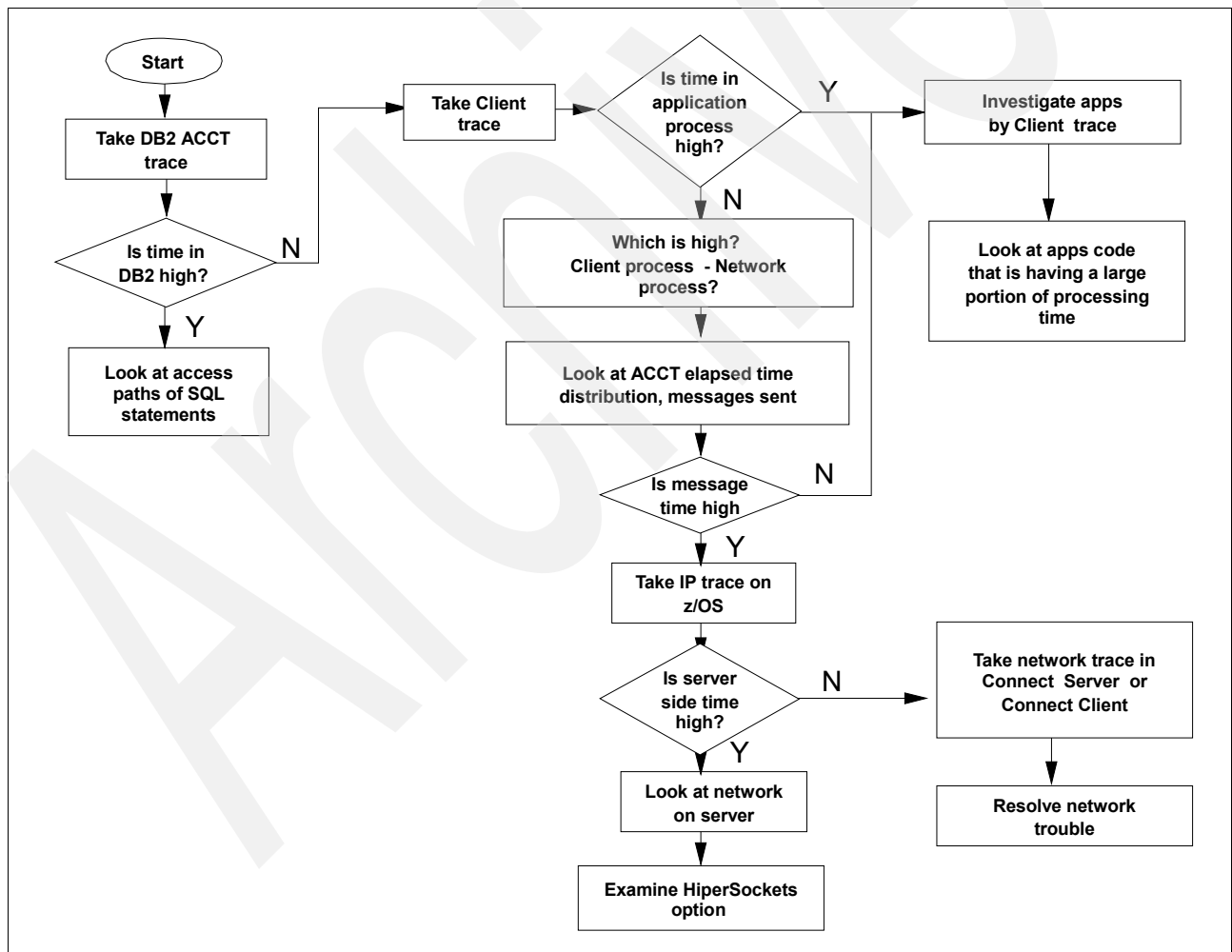


Figure 8-27 Flowchart describing the problem determination method in n-tier environment

Figure 8-27 on page 390 illustrates the following steps:

1. When a problem is first reported, step 1 is to take an accounting trace. The accounting trace will give the summary of the time spent in DB2 and the time outside DB2, which includes the network, DB2 Connect Server, and the time required to run the application in the application server.
2. If we notice that most of the time is spent in DB2, then we have to look at the SQL and the access path of the SQL. It is a good idea to look at the record trace to see the access path at runtime. If the elapsed time (ET) is in DB2, at this stage the problem is already identified to be SQL in the application or the application design and this needs to be fixed.
3. If the elapsed time (ET) is outside DB2, take a client trace of the application.
4. If the client trace indicates that the application processing is high, look at the application design and the application code. A client (Java) trace may help to determine if the application process is in a loop.
5. If the client trace indicates a high driver time or high client processing time, then a closer look at the DB2 accounting report on the distribution statistics is helpful. The DDF section of the report gives the number of messages sent and received by the client. It also gives statistics as to whether blocking was used. If blocking is not used, it translates into high time spent in communication.
6. If the number of messages is high, see whether the application design can make use of stored procedures or can be converted to a static model such as in SQLJ.
7. If the application design is satisfactory, then consider taking an IP trace. The IP trace determines whether the problem is at the server or outside the server.
8. If the problem is at the server, then, with the help of network engineers and system programmers, determine the nature of the problem and fix it. Consider using HiperSockets as an alternative to speed up network calls.
9. If the problem is outside the server, consider using the network analyzer or sniffer devices to see if the outside network is robust.

The above steps discuss the flow for problem determination. It is recommended that proper benchmarks be done in a production environment so that when there are performance issues, the results (ET) can be compared to determine whether they are under the defined threshold.

Archived

## Appendixes

In this part we have included the following Appendixes:

- ▶ Appendix A, “DRDA-related maintenance” on page 395
- ▶ Appendix B, “Configurations and workload” on page 401
- ▶ Appendix C, “Sample applications” on page 419
- ▶ Appendix D, “Sample programs for performance analysis” on page 443

Archived

## DRDA-related maintenance

In this appendix, we look at recent maintenance for DB2 9 that generally relates to DB2 distributed functions.

We present lists of APARs that are applicable to DB2 9 and DB2 V8. The APARs represent a snapshot of the current maintenance for performance and availability functions at the time of writing. As such, the list will become incomplete or even incorrect at the time of reading. It is here to identify areas of performance or functional improvements.

Make sure that you contact your IBM Service Representative for the most current maintenance at the time of your installation. Also check on RETAIN for the applicability of these APARs to your environment, as well as to verify pre- and post-requisites.

We recommend that you use the Consolidated Service Test (CST) as the base for service.

## A.1 Recent DRDA APARs

We present a list of DRDA-related APARs in Table A-1.

Table A-1 DB2 9 current DRDA-related APARs

APAR #	Area	Text	PTF and notes
II14203	info	TCP/IP and shared memory	See the list inside
PK03045	WLM	Clients will now receive server weights that take into account whether or not a DB2 member of a data sharing group is resource constrained	UK09411 (V8)
PK05198	Sysplex workload balancing	Data sharing migration to V8 while in a coexistence environment when using DB2 Connect connection concentrator in compatibility mode	UK06864 (V8)
PK06697	Sysplex workload balancing	Data sharing migration to V8 while in a coexistence environment when using DB2 Connect connection concentrator in new-function mode	UK12978 (V8)
PK18454	zIIP	Exploitation of the IBM System z9 Integrated Information Processor (IBM zIIP) for DRDA threads	UK15499 (V8) see also OMEGAMON PE APAR PK25395
PK25395	zIIP	OMEGAMON XE for DB2 Performance Expert on z/OS will be modified to provide monitoring support for zIIP	UK15518 (V8)
PK30450	Sysplex workload Balancing	JCC T4 driver with enable SYSPLEXWLB=TRUE receives conversation dellocation with DB2 z/OS server	UK21949 (V8)
PK38867	zIIP	WLM sysplex routing services of z/OS V1R9.0 services were enhanced to include an awareness of the available capacity of a server running with zIIP specialty engines.	UK35518/9 (V8/V9)
PK41236	Sysplex workload balancing	IBM Data Server Driver for JDBC and SQLJ (version 3.51.x/ 4.1.x or higher) type 4 connectivity, with the "keepDynamic" and "enableSysplexWLB" DB2BaseDataSource properties enabled, are used to access a DB2 for z/OS data sharing group.	UK40814/5 (V8/V9) PE see PK89820
PK41661	SQL interrupts support	"SQL Interrupt" processing was introduced in DB2 z/OS V8 to provide a mechanism to interrupt only the currently executing SQL, resulting in a negative SQLCODE. The DSNZPARM DSN6FAC macro service has been changed to provide a new SQLINTRP keyword to disable it.	UK26033 (V8)
PK44617	Trusted context	Implementation of needed functions: <ul style="list-style-type: none"> <li>► Managing trusted context users in RACF.</li> <li>► Use of a wild card character in the JOBNAME attribute. Use of a role value as default for CURRENT SCHEMA and CURRENT PATH special registers.</li> <li>► Trace record is now written when the ROLE keyword is specified on the START TRACE command.</li> </ul>	UK33449 (V9)



APAR #	Area	Text	PTF and notes
PK46079	Progressive streaming	Configuration setting to establish default Progressive Streaming behavior for the JDBC Universal driver. The DSNZPARM DSN6FAC macro service has been changed to provide a new PRGSTRIN keyword.	UK26257 (V9) see also APAR IY99846 for JCC
PK47649	IFCID 001	This PTF adds z/OS metrics CPU % in four granularities (LPAR, DB2 subsystem,MSTR, DBM1) to IFCID 001 in a new data section (#13).	UK32198 (V8)
PK50575	zAAP accounting	Support for reporting for redirection on an IBM zAAP processor (XML)	UK36263
PK51045	OMEGAMON PE V4.1	Provide missing support of some Specialty Engine counters (report on zIIP and zAAP)	UK37241
PK56287	256-bit AES encryption support	<p>DB2 for z/OS supports AES requests from remote clients.</p> <ul style="list-style-type: none"> <li>▶ DB2 for z/OS V8 CM users: The installation of this DB2 maintenance will allow DB2 to tolerate AES encrypted security tokens from remote clients that also support AES. However, DB2 will not provide any functionality to process the AES encrypted security token and DES encryption will continue to be used.</li> <li>▶ DB2 for z/OS V8 NFM and DB2 9 for z/OS CM users: The installation of this DB2 maintenance will enable the support for AES decryption processing of the userid and/or password during remote connection processing. Only server support is being provided.</li> <li>▶ DB2 9 for z/OS NFM users: The installation of this DB2 maintenance will enable the support for AES encryption and decryption processing of the userid and/or password during remote connection processing. Requester and server support is being provided.</li> </ul>	UK38142/3 (V8/V9) You need also ICSF APAR OA27145 (PTF UA45350)
PK56356	IFCID 001	This PTF adds z/OS metrics CPU % in four granularities (LPAR, DB2 subsystem ,MSTR, DBM1) to IFCID 001 in a new data section (#13).	UK33795 (V9)
PK59385	SQL Interrupt	New system parameter to disable DDF support of SQL Interrupt processing	UK33843 (V9)
PK62161	IFCID02, IFCID03	Added multi-row fetch/insert information in DB2 statistics (IFCID 2) and DB2 accounting (IFCID 3) records. OMEGAMON PE support with PK89128.	UK44050/1 (V8/V9)
PK64298	DB2-supplied stored procedures	Reduce the size of the XML output document for GET_SYSTEM_INFO, and to improve performance, allow the user to select which information to return. SYSPROC.GET_CONFIG, SYSPROC.GET_MESSAGE, and SYSPROC.GET_SYSTEM_INFO	UK37310/1 (V8/V9)
PK64045	Private Protocol	V8/V9 Changes for private protocol removal	OPEN (9/9/30)

APAR #	Area	Text	PTF and notes
PK65367	TCP/IP 2PC	New function is provided to externalize the extended CRRTKN information in the Display Thread command report and in trace records.	UK39483 (V9)
PK67794	DB2 supplied stored procedures	This new function APAR ships function enhancements for the stored procedures SYSPROC.ADMIN_COMMAND_DB2 and SYSPROC.GET_CONFIG; ships a new stored procedure SYSPROC.ADMIN_INFO_SYSPARM; and updates the stored procedures SYSPROC.ADMIN_INFO_HOST, SYSPROC.GET_MESSAGE and SYSPROC.GET_SYSTEM_INFO to support the changes made to GET_CONFIG and ADMIN_COMMAND_DB2.	UK46487/8(V8/V9)
PK68746	Performance when using IBM Data Server Driver for JDBC and SQLJ Type 4	Performance improvement for applications using the IBM Data Server Driver for JDBC and SQLJ Type 4 Connectivity client with autocommit enabled when connected to DB2 for z/OS server.	UK42748/9 (V8/V9)
PK69659	direct XA support	XA support for the IBM Data Server for CLI and ODBC driver introduced in V9.5 for FixPack 3	UK43747/8 (V8/V9)
PK74330	DB2 supplied stored procedures	New procedure SET_CLIENT_INFO	UK46551/2 (V8/V9)
PK75338	DB2DATB member	When filtering a space list to just indexspaces, a large number of the spaces are not visible	UK42800
PK76143	Encryption	DB2 z/OS SERVER_ENCRYPT authentication enforcement. New DB2 function is required to validate that remote client systems can only access a DB2 z/OS server via encrypted forms of security	OPEN
PK78553	The private to DRDA protocol REXX tool	When you run the DSNTP2DP tool, in addition to the status messages, it generates three types of output: the outalias output, the outpkgs output, and the outplans output. DSNTP2DP does not give complete output in the outplans component.	UK46942 (V9)
PK79228	Data sharing	DB2 Version 9 for z/OS introduces a new feature: randomization of group attachment requests. The randomization algorithm is improved to more evenly distribute group attachment requests. This PTF also adds a new DB2 subsystem parameter, RANDOMATT=YES/NO, which may be specified for each DB2 subsystem within the group. The default setting remains YES, which is assumed if no action is taken to update the DB2 subsystem parameters.	UK44898 (V9)
PK79327	Data sharing	Companion to PK79228	UK44899 (V9)
PK79700	IFCID 247	The problem occurs in a distributed environment when a statement, containing an input host variable, is executed on a DB2 server with a DB2 trace active to trace input host variables (IFCID 247).	UK45418/9 (V8/V9)

APAR #	Area	Text	PTF and notes
PK80474	Display DDF	<p>Display DDF command report to provide the following new information:</p> <ul style="list-style-type: none"> <li>▶ Location Alias information, as defined in the BSDS DDF record via the DSNJU003 Change Log inventory utility DDF Alias statement, for DB2 z/OS V8 users.</li> <li>▶ The IP address associated to the entire location.</li> <li>▶ Addition of DSNL089I</li> </ul>	UK44299/300 (V8/V9)
PK81175	SSL	Resolved connection issues	UK45981
PK85150	DB2 Driver for JDBC and SQLJ	IBM DB2 Driver for JDBC and SQLJ version 3.52.112 is provided by this APAR (JCCV352112).	UK46727
PK89128	OMEGAMON PE	Multi-row information editing and other corrections for V4.2	UK47981
PK89820	IBM Data Server driver for JDBC and SQLJ (version V9.1 FP3 or higher)	Adds enhancements that PK41236 provides to versions of DB2 for LUW below V9.5 FP3.	UK48537 also V8

Archived



## Configurations and workload

This appendix shows the configurations of the different platform environment used during this project and the TRADE application workload.

## B.1 Configurations

This section describes the environment in which we performed our tests. Figure B-1 shows our starting z/OS and AIX configurations. We used four DB2 9 for z/OS subsystems. DB9A was a standalone subsystem in LPAR SC63. D9C1, D9C2, and D9C3 were data sharing members on LPARs SC63, SC64 and SC70, respectively. The processors on WTSCPLX2 were shared across all three LPARs except SC64, which did not have the zIIP and zAAP processors defined.

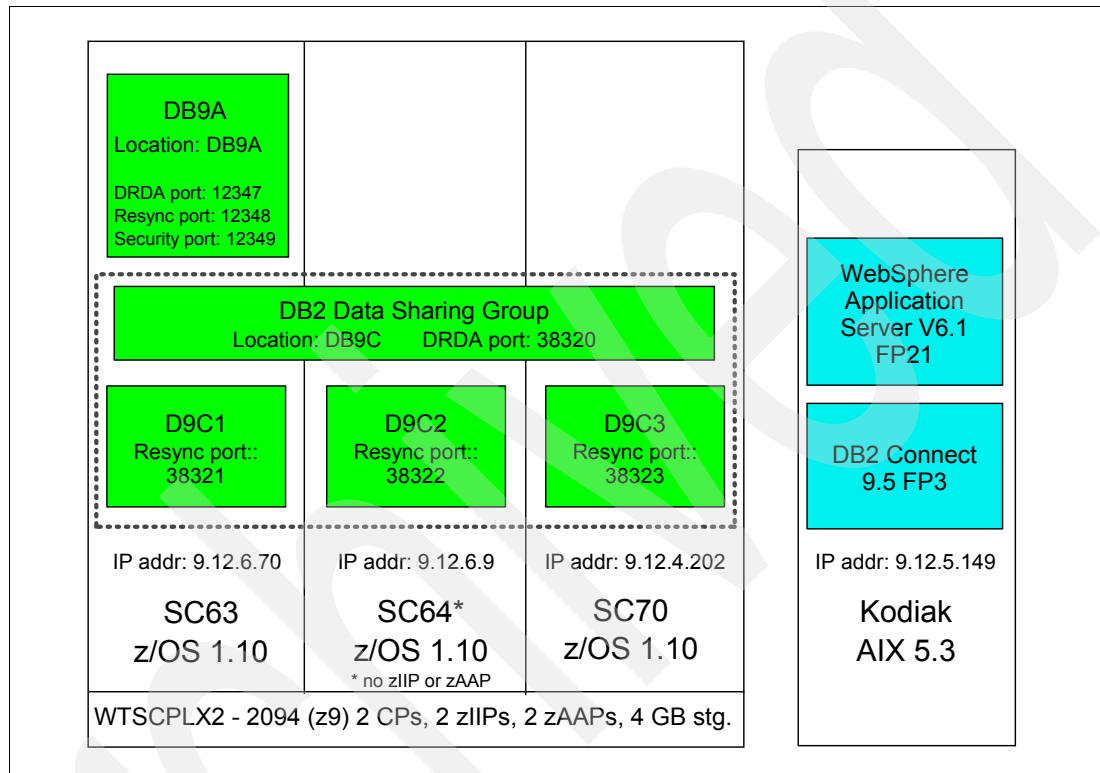


Figure B-1 Starting z/OS and AIX configurations

In our starting configuration we only used the IP addresses for the LPARs. All of our DB2s listened on any IP address (INADDR\_ANY). Only DB9A defined a security port to support Secure Socket Layer (SSL).

Figure B-2 on page 403 shows our configuration after we implemented Sysplex Distributor and DVIPA for our DB2 data sharing group. Because we defined the Group DVIPA and member DVIPAs in the BSDS for the data sharing members, all the DB2s could still listen on any IP address (INADDR\_ANY).

Figure B-2 also shows the addition of a Linux for IBM System z, in a z/VM® LPAR. We used this to implement HiperSocket support.

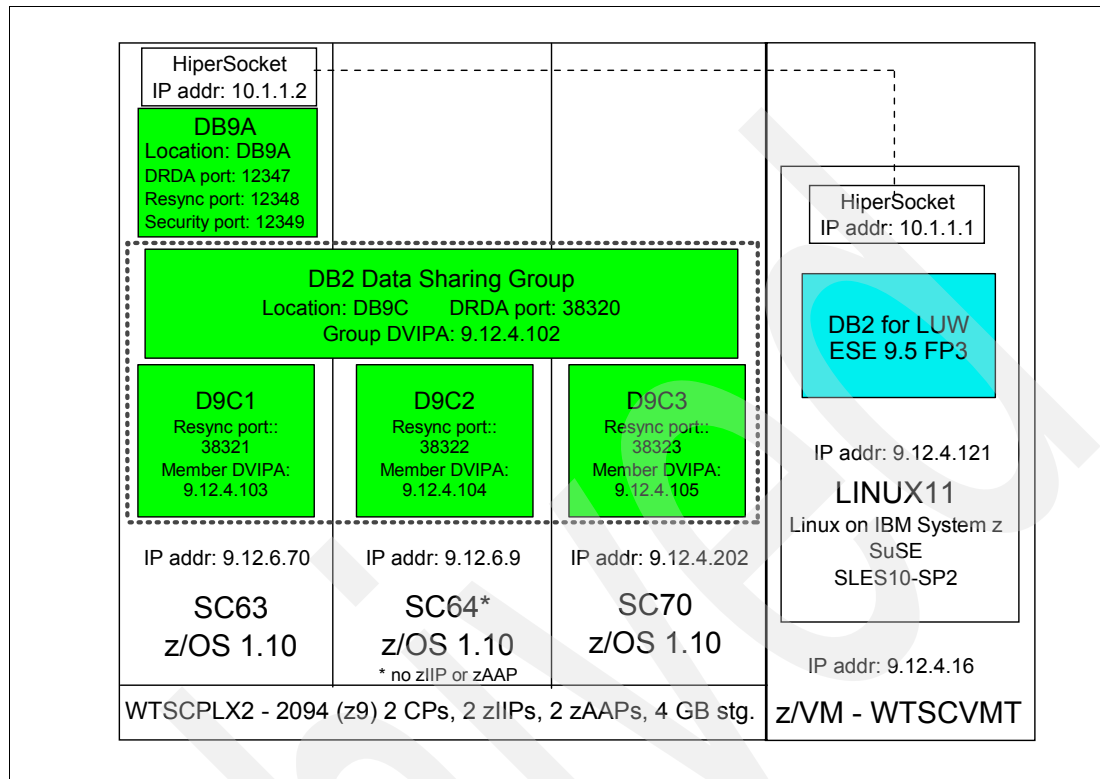


Figure B-2 Configuration after implementing DVIPA support, plus Linux on IBM System z.

Figure B-3 shows the addition of LOCATION subsets in our configuration.

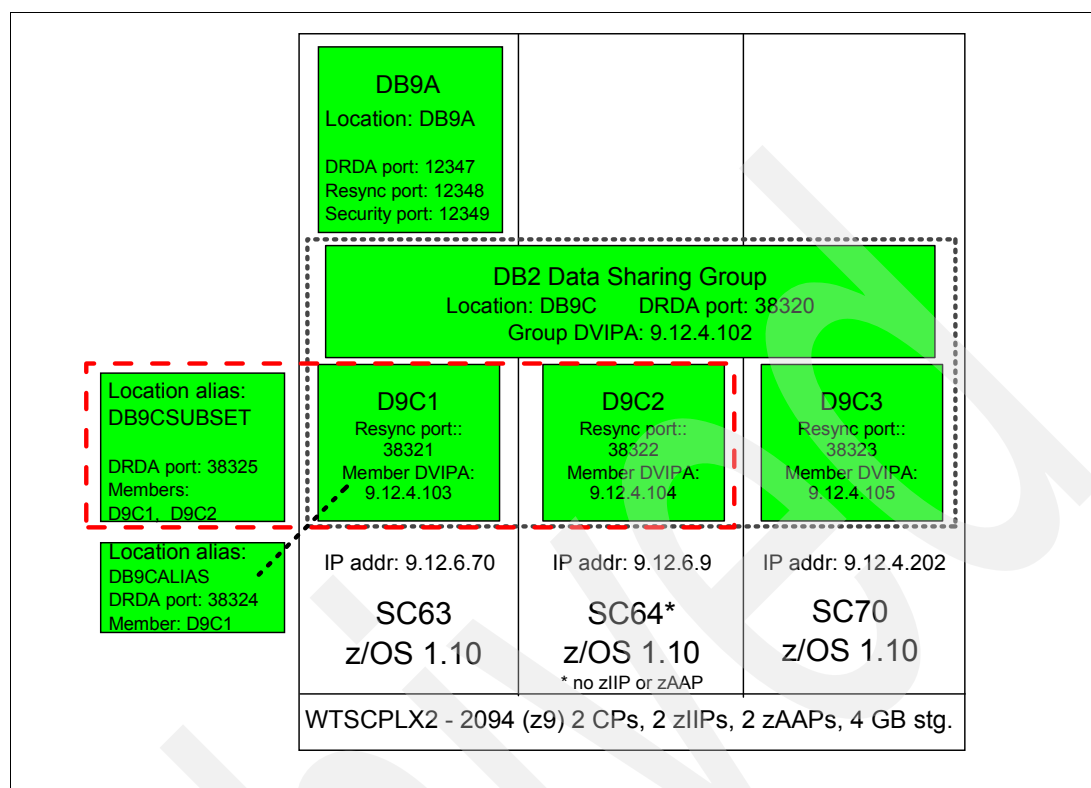


Figure B-3 Location subsets in our DB2 data sharing group.

Members D9C1 and D9C2 comprise location subset DB9CSUBSET. Member D9C1 is the only member in subset DB9CALIAS.

## B.2 The TRADE workload

In this appendix we provide a description of the TRADE workload which was used for deployment verification in 6.1, “High availability aspects of DRDA access to DB2 for z/OS” on page 234.

This appendix contains the following sections:

- ▶ B.2.1, “IBM Trade performance benchmark sample for WebSphere Application Server V6.1” on page 404.
- ▶ B.2.2, “TRADE installation” on page 405.

### B.2.1 IBM Trade performance benchmark sample for WebSphere Application Server V6.1

The WebSphere performance benchmark sample provides a suite of IBM-developed workloads for characterizing performance of the WebSphere Application Server. The workloads consist of an end-to-end Web application and a full set of primitives. The applications are a collection of Java classes, Java Servlets, Java Server Pages, Web Services, and Enterprise Java Beans built to open J2EE APIs. Together these provide



versatile and portable test cases designed to measure aspects of scalability and performance. See Figure B-4.

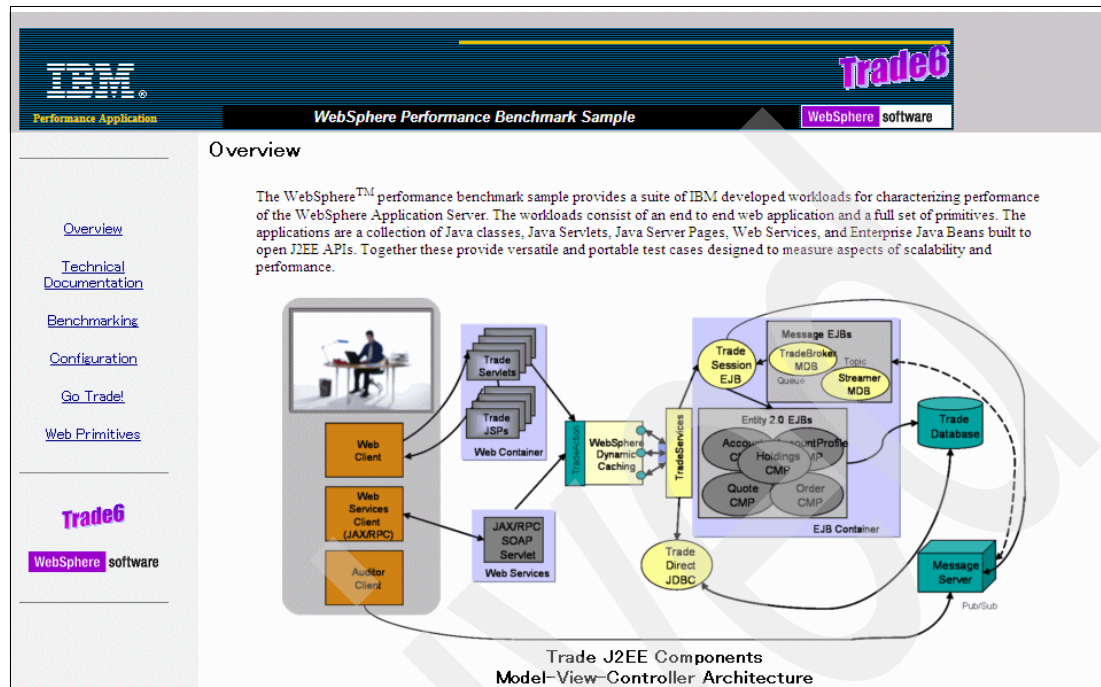


Figure B-4 Trade overview

### B.2.1.1 Our environment

The environment consists of the following elements:

- ▶ AIX Version 5.3
- ▶ WebSphere Application Server V6.1 Fixpack 21
- ▶ DB2 Connect V9.5 Fixpak 3

**Note:** The steps show the installation of trade application assuming software was installed.

## B.2.2 TRADE installation

Download your copy of trade application from the following Web page:

[http://www14.software.ibm.com/webapp/download/preconfig.jsp?id=2005-06-07+07%3A20%3A48.610697R&cat=webserver&fam=&s=&S\\_TACT=104CBW71&S\\_CMP=&st=2&sp=20](http://www14.software.ibm.com/webapp/download/preconfig.jsp?id=2005-06-07+07%3A20%3A48.610697R&cat=webserver&fam=&s=&S_TACT=104CBW71&S_CMP=&st=2&sp=20)

Extract the files from the downloaded package.

### B.2.2.1 DB2 for z/OS

Create the Trade database by performing the following steps. The skeleton JCL is prepared in the trade package.

1. The trade6db\_package.jcl file contained in the tradeinstall/zOS directory, upload them to a data set in your TSO environment in your z/OS system.

**Note:** The trade6db\_package.jcl is stored in EBCDIC format and so no conversions are necessary.

2. Customize the JCL by following the instructions in the JCL.
3. Submit the job to create the table spaces, tables, and indexes for the Trade database.

### B.2.2.2 AIX box

The trade application installation script will define and install necessary resources in your installation of WebSphere Application Server. The script provides the following install options:

- ▶ DB2 for LUW (T4 Driver)
- ▶ DB2 for z/OS (T2 Driver)
- ▶ Oracle

**Tip:** You cannot choose an option to use T4 Driver for DB2 for z/OS. But you have to choose DB2 for z/OS or your application will not run. After running installation script, you can modify your data source setting from the WebSphere Application Server administrative console. We will show how to change them later.

Example B-1 on page 407 illustrates running the configuration and installation script in our AIX environment. Prepare the following information before you start. You will be prompted to input them:

- ▶ Username  
User ID used to install trade application, in our example we used “wsadmin”
- ▶ Password  
Password for username
- ▶ WebSphere Application Server installation  
If you are using global security or/and cluster installation, you will be prompted.
- ▶ WebSphere Application Server node  
Install nodes for trade application. You will get lists of nodes to choose from. In our example we used “kodiakNode01”.
- ▶ Server  
Install server for trade application. You will get lists of servers to choose from. In our example we used “server1”.
- ▶ Backend DB type  
Choose from db2, oracle, or db2zos. You must choose “db2zos” when running against DB2 for z/OS server.

► DB driver path

Type in path for your T4 driver path. In our installation,  
<instance\_home>/sqllib/java/db2jcc.jar and  
<instance\_home>/sqllib/java/db2jcc\_lisence\_cisuz.jar. In our example, we used  
"/home/db2inst1/sqllib/java/db2jcc.jar " and  
"/home/db2inst1/sqllib/java/db2jcc\_lisence\_cisuz.jar".

► DB name

Location name of DB2 for z/OS. In our installation, "DB9C" is location name for DB2 data sharing group.

► DB username

User ID used to connect to DB2 for z/OS. In our installation, "PAOLOR7" was used.

► DB password

Password associated with DB username.

*Example B-1 Running the configuration and installation script in AIX*

---

```
# /usr/IBM/WebSphere/AppServer/bin/wsadmin.sh -f trade.jacl
Realm/Cell Name: <default>
Username: wsadmin
Password:
WASX7209I: Connected to process "dmgr" on node kodiakCellManager01 using SOAP c
connector; The type of process is: DeploymentManager
trade.jacl
resource_scripts.jacl
```

-----  
Trade Install/Configuration Script

Operation: all  
Silent: false  
-----

Global security is (or will be) enabled (true|false) [false]:  
**false**

Is this a cluster installation (yes|no) [no]:  
**no**  
-----

Collecting Single Server or Managed Server Info  
-----

Available Nodes:  
kodiakCellManager01  
kodiakNode01

Select the desired node [kodiakCellManager01]:  
**kodiakNode01**

Available Servers:  
dmgr  
nodeagent  
server1

Select the desired server [dmgr]:  
**server1**  
-----

#### Collecting Database/Datasource Information

Select the backend database type (db2|oracle|db2zos) [db2]:  
**db2zos**

NOTE: wsadmin requires ";" for delimiting the database driver path regardless of platform!

Please enter the database driver path [c:/sqllib/java/db2jcc.jar;c:/sqllib/java/db2jcc\_license\_cu.jar;c:/sqllib/java/db2jcc\_license\_cisuz.jar]:  
**/home/db2inst1/sqllib/java/db2jcc.jar;/home/db2inst1/sqllib/java/db2jcc\_license\_cisuz.jar**

Please enter the driver native library path [/usr/lpp/db2/db2810/jcc/lib]:

Please enter the database name (location) [tradedb]:  
**db9c**

Please enter the database username [db2admin]:  
**paolor7**

Please enter the database password [password]:  
**newpswd**

#### Configuring JDBC/Datasource Resources

Scope: kodiakNode01(cells/kodiakCell101/nodes/kodiakNode01|node.xml#Node\_1)

Creating JAAS AuthData TradeDataSourceAuthData...  
TradeDataSourceAuthData already exists!

Creating JDBC Provider DB2 Universal JDBC Driver Provider...

Provider Name: DB2 Universal JDBC Driver Provider

Implementation Class: com.ibm.db2.jcc.DB2ConnectionPoolDataSource

Classpath: /home/db2inst1/sqllib/java/db2jcc.jar;/home/db2inst1/sqllib/java/db2jcc\_license\_cisuz.jar

XA enabled: false

DB2 Universal JDBC Driver Provider created successfully!

Creating DataSource TradeDataSource...

Datasource Name: TradeDataSource

JNDI Name: jdbc/TradeDataSource

Statement Cache Size: 60

Database Name: db9c

Creating Datasource properties...

Creating Connection Pool using defaults...

Creating Connection Factory...

TradeDataSource created successfully!

JDBC Resource Configuration Completed!!!

#### Configuring JMS Resources

Scope: kodiakNode01(cells/kodiakCell101/nodes/kodiakNode01|node.xml#Node\_1)

-----  
Creating JAAS AuthData TradeOSUserIDAuthData...  
TradeOSUserIDAuthData already exists!

Creating SIBus kodiakNode01...  
kodiakNode01 already exists!

Adding SIBus member kodiakNode01 - server1...  
Default DataSource: true  
Bus member already exists!

Creating SIB Destination TradeBrokerJSD...  
TradeBrokerJSD already exists!

Creating SIB Destination Trade.Topic.Space...  
Trade.Topic.Space already exists!

Creating JMS Queue Connection Factory TradeBrokerQCF...  
TradeBrokerQCF already exists!

Creating JMS Topic Connection Factory TradeStreamerTCF...  
TradeStreamerTCF already exists!

Creating JMS Queue TradeBrokerQueue...  
TradeBrokerQueue already exists!

Creating JMS Topic TradeStreamerTopic...  
TradeStreamerTopic already exists!

Creating MDB Activation Spec TradeBrokerMDB...  
TradeBrokerMDB already exists!

Creating MDB Activation Spec TradeStreamerMDB...  
TradeStreamerMDB already exists!

-----  
JMS Resource Configuration Completed!!!  
-----

Saving...

-----  
Installing Trade  
-----

Installing application {Trade}...  
Application Name: Trade  
Ear file: trade.ear  
Target Node: kodiakNode01  
Target Server: server1  
Deploy EJB: true  
Deploy WebServices: true  
Use default bindings: true  
Use Ear MetaData: true  
Deployed DB Type: DB2UDBOS390\_V8

Starting application install...  
WASX7327I: Contents of was.policy file:  
grant codeBase "file:\${application}" {  
permission java.security.AllPermission;

```

};
ADMA5016I: Installation of Trade started.
ADMA5058I: Application and module versions are validated with versions of deployment
targets.
ADMA5018I: The EJBDeploy command is running on enterprise archive (EAR) file
/tmp/app10631.ear.
Starting workbench.
EJB Deploy configuration directory: /usr/IBM/WebSphere/AppServer/profiles/Dmgr01
/ejbdeploy/configuration/
framework search path: /usr/IBM/WebSphere/AppServer/deploytool/itp/plugins
Debug options:
Creating the project.
Deploying jar tradeEJB
Creating Top Down Map
Generating deployment code
Generating DDL
Generating DDL
ejbModule/com/ibm/websphere/samples/trade/util/MDBStats.java(21): The serializable class
MDBStats does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/soap/TradeWebSoapProxy.java(27): The field
TradeWebSoapProxy.servicePort is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeStreamerMDB.java(17): The serializable
class TradeStreamerMDB does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeStreamerMDB.java(21): The field
TradeStreamerMDB.mdc is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeStreamerMDB.java(22): The field
TradeStreamerMDB.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeStreamerMDB.java(24): The field
TradeStreamerMDB.tradeHome is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeBrokerMDB.java(20): The serializable
class TradeBrokerMDB does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeBrokerMDB.java(25): The field
TradeBrokerMDB.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/TradeBean.java(22): The serializable class
TradeBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/ejb/QuoteBean.java(22): The field Quot
eBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/OrderBean.java(24): The field Orde
rBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/KeySequenceBean.java(19): The serializable
class KeySequenceBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/ejb/KeySequenceBean.java(21): The field
KeySequenceBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/KeyGenBean.java(22): The field
KeyGenBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/HoldingBean.java(45): The method
getQuoteFromSymbol(String) from the type HoldingBean is never used locally
ejbModule/com/ibm/websphere/samples/trade/ejb/AccountProfileBean.java(18): The field
AccountProfileBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/ejb/AccountBean.java(25): The field
AccountBean.context is never read locally
ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1077): The method
getAccountDataForUpdate(int, Connection) from the type TradeDirect is never used locally
ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1088): The method
getQuoteData(String) from the type TradeDirect is never used locally
ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1163): The method
getOrderData(int) from the type TradeDirect is never used locally
ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1244): The method
getAccountProfileData(Integer) from the type TradeDirect is never used locally

```

ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1380): The method  
 updateQuoteVolume(Connection, QuoteDataBean, double) from the type TradeDirect is never  
 used locally  
 ejbModule/com/ibm/websphere/samples/trade/direct/TradeDirect.java(1029): The  
 method getAccountDataForUpdate(Connection, String) from the type TradeDirect is  
 never used locally  
 ejbModule/com/ibm/websphere/samples/trade/command/UpdateQuotePriceCommand.java(16): The  
 serializable class UpdateQuotePriceCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/UpdateAccountProfileCommand.java(15): The  
 serializable class UpdateAccountProfileCommand does not declare a static final  
 serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/SellCommand.java(15): The serializable  
 class SellCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/ResetTradeCommand.java(15): The  
 serializable class ResetTradeCommand does not declare a static final serialVersionUID field  
 of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/RegisterCommand.java(16): The  
 serializable class RegisterCommand does not declare a static final serialVersionUID field  
 of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/QuoteCommand.java(15): The serializable  
 class QuoteCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/OrdersCommand.java(16): The serializable  
 class OrdersCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/OrderCompletedCommand.java(17): The  
 serializable class OrderCompletedCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/MarketSummaryCommand.java(15): The  
 serializable class MarketSummaryCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/LogoutCommand.java(15): The serializable  
 class LogoutCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/LoginCommand.java(15): The serializable  
 class LoginCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/HoldingsCommand.java(16): The  
 serializable class HoldingsCommand does not declare a static final serialVersionUID field  
 of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/HoldingCommand.java(15): The serializable  
 class HoldingCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/CreateQuoteCommand.java(16): The  
 serializable class CreateQuoteCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/ClosedOrdersCommand.java(17): The  
 serializable class ClosedOrdersCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/BuyCommand.java(15): The serializable  
 class BuyCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/AllQuotesCommand.java(17): The  
 serializable class AllQuotesCommand does not declare a static final serialVersionUID field  
 of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/AccountProfileCommand.java(15): The  
 serializable class AccountProfileCommand does not declare a static final serialVersionUID  
 field of type long  
 ejbModule/com/ibm/websphere/samples/trade/command/AccountCommand.java(15): The serializable  
 class AccountCommand does not declare a static final serialVersionUID field of type long  
 ejbModule/com/ibm/websphere/samples/trade/TradeConfig.java(189): The field Trade  
 Config.NAMESERVICE\_TYPE\_PROPERTY is never read locally  
 ejbModule/com/ibm/websphere/samples/trade/TradeConfig.java(191): The field Trade  
 Config.NAMESERVICE\_PROVIDER\_URL\_PROPERTY is never read locally

```

ejbModule/com/ibm/websphere/samples/trade/RunStatsDataBean.java(15): The serializable class
RunStatsDataBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/QuoteDataBean.java(16): The serializable class
QuoteDataBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/OrderDataBean.java(20): The serializable class
OrderDataBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/MarketSummaryDataBeanWS.java(25): The
serializable class MarketSummaryDataBeanWS does not declare a static final serialVersionUID
field of type long
ejbModule/com/ibm/websphere/samples/trade/MarketSummaryDataBeanWS.java(36): The field
MarketSummaryDataBeanWS.gainPercent is never read locally
ejbModule/com/ibm/websphere/samples/trade/MarketSummaryDataBean.java(17): The serializable
class MarketSummaryDataBean does not declare a static final serialVersionUID field of type
long
ejbModule/com/ibm/websphere/samples/trade/HoldingDataBean.java(17): The serializable class
HoldingDataBean does not declare a static final serialVersionUID field of type long
ejbModule/com/ibm/websphere/samples/trade/AccountProfileDataBean.java(14): The serializable
class AccountProfileDataBean does not declare a static final serialVersionUID field of type
long
ejbModule/com/ibm/websphere/samples/trade/AccountDataBean.java(17): The serializable class
AccountDataBean does not declare a static final serialVersionUID field of type long
Invoking RMIC.
Writing output file
Shutting down workbench.
EJBDeploy complete.
0 Errors, 53 Warnings, 0 Informational Messages
ADMA5007I: The EJBDeploy command completed on
/usr/IBM/WebSphere/AppServer/profiles/Dmgr01/wstemp/wstemp/app_120edb8de1d/dp1/dp1_Trade.ear
WSWS0041I: Web services deploy task completed successfully.
ADMA5005I: The application Trade is configured in the WebSphere Application Server
repository.
ADMA5053I: The library references for the installed optional package are created.
ADMA5005I: The application Trade is configured in the WebSphere Application Server
repository.
ADMA5001I: The application binaries are saved in
/usr/IBM/WebSphere/AppServer/profiles/Dmgr01/wstemp/Script120edb69c9c/workspace/cells/kodia
kCell01/applications
/Trade.ear/Trade.ear
ADMA5005I: The application Trade is configured in the WebSphere Application Server
repository.
SECJ0400I: Successfully updated the application Trade with the appContextIDForSecurity
information.
ADMA5011I: The cleanup of the temp directory for application Trade is complete.
ADMA5013I: Application Trade installed successfully.
Install completed successfully!

-----
Trade Installation Completed!!!
-----

Saving...

Saving config...
#

```

---



**Tip:** If you have failed to complete running your scripts, check your WebSphere Application Server administrative console if any of configurations or installation are done. If you are going to re-run your script, you have to delete manually what is configured or installed from your last installation.

After you are done with running scrips, go to your WebSphere Application Server administrative console. The trade application should be installed but it may not be started, as shown in Figure B-5. You do not need to start trade application at this time.

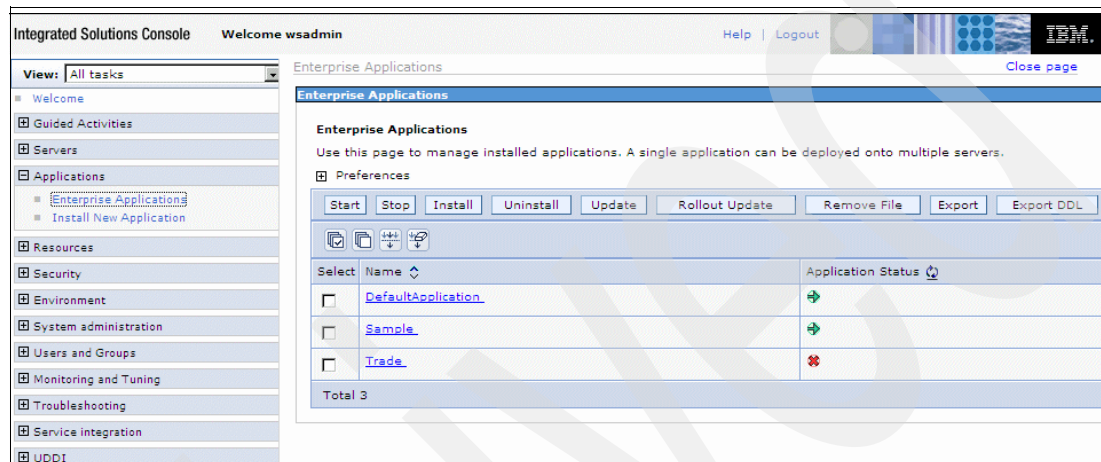


Figure B-5 WebSphere Application Server admin console after installation

Under **Resources** → **JDBC** → **JDBC Provider**, you should see new JDBC provider defined, as shown in Figure B-6.

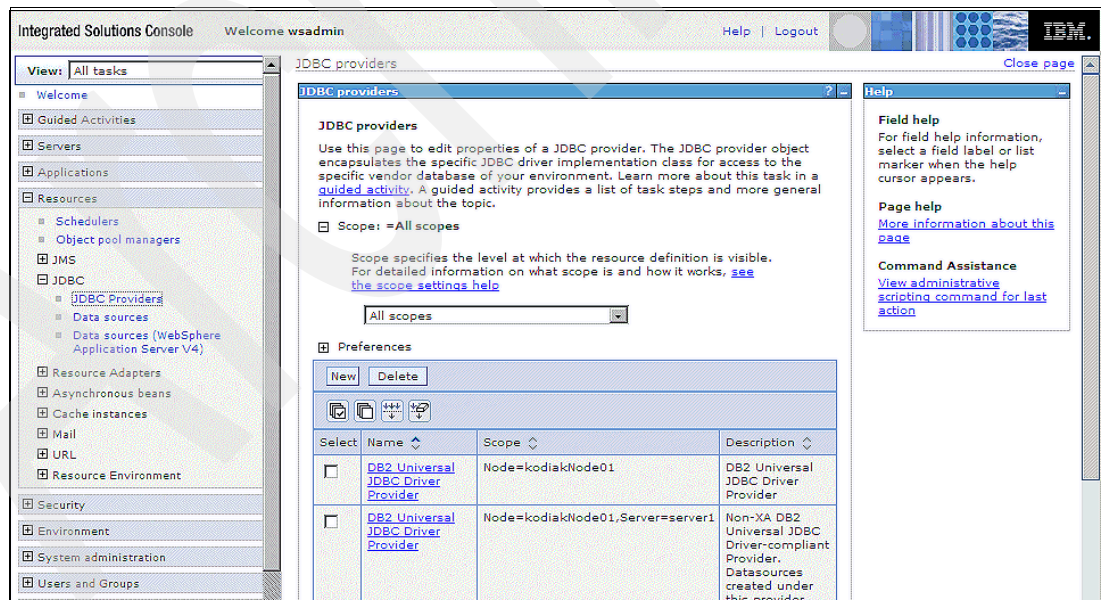


Figure B-6 JDBC Provider defined from configuration script

Customize your datasource settings so you can connect to DB2 for z/OS through the network, where the default installation for “db2zos” will be T2 Driver. In our example, we have also done the set up for sysplex workload balancing.

Navigate to **Resources** → **JDBC** → **Datasource**, you will see lists of datasource defined in your installation. You should see “TradeDataSource”, if not check your scope match with a node you specified at script or “All scopes”. Figure B-7 shows datasource for our example.

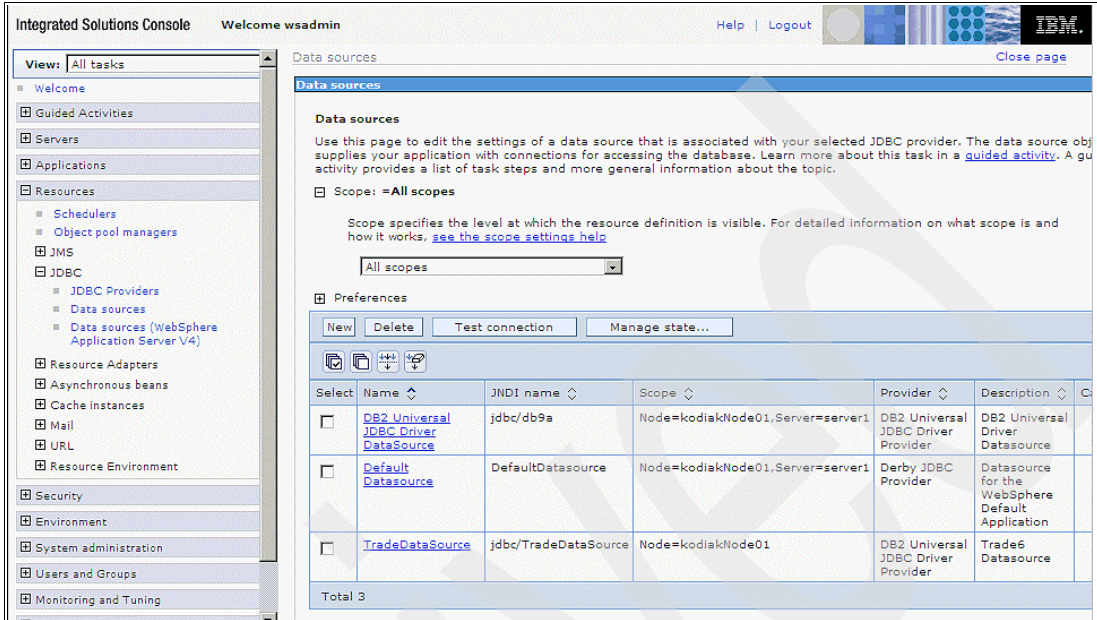


Figure B-7 TradeDataSource from WebSphere Application Server admin console

Click **TradeDataSource** for settings for your trade application datasource. At the bottom, you will find the Driver type, Server name, and Port number. Change Driver type to “4”, Server name to “IP address” or the domain name for your DB2 for z/OS, and change the Port number to the DRDA service port. In our example, settings will look like Figure B-8. In addition, we added two properties related to sysplex workload balancing in datasource custom properties. Refer to Chapter 6, “Data sharing” on page 233 for more information.

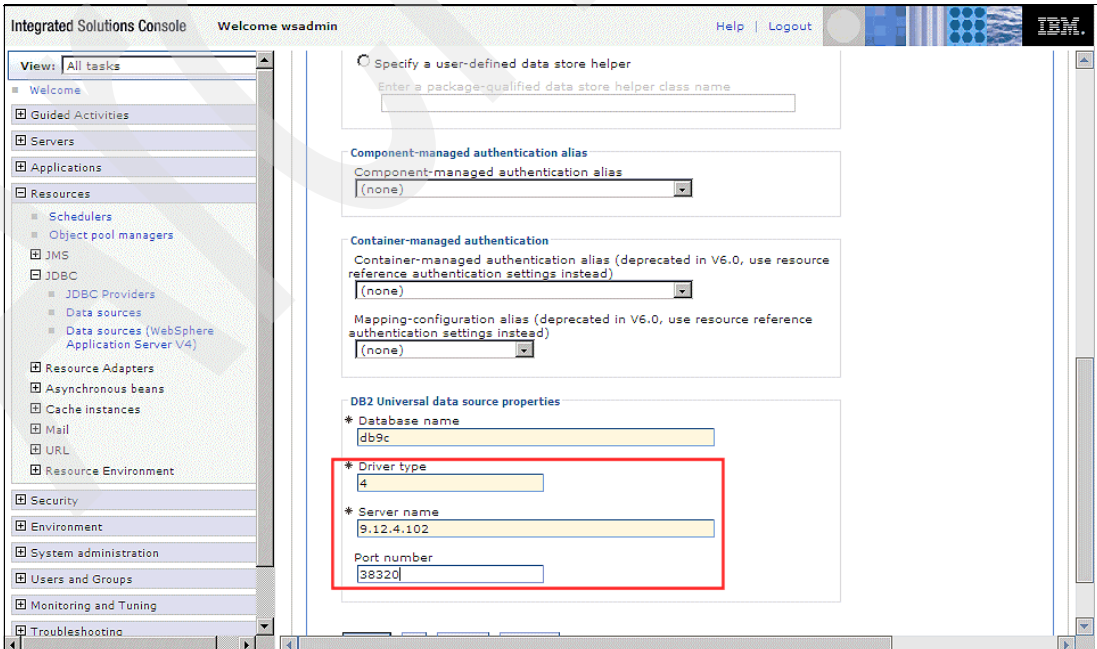


Figure B-8 Modify DataSource to Type 4 Connection

Go to **Servers** → **Application servers**, and restart the application server in which the trade application is installed. The panel should look like Figure B-9. After restarting, navigate to **Applications** → **Enterprise Application** to verify trade application is started.

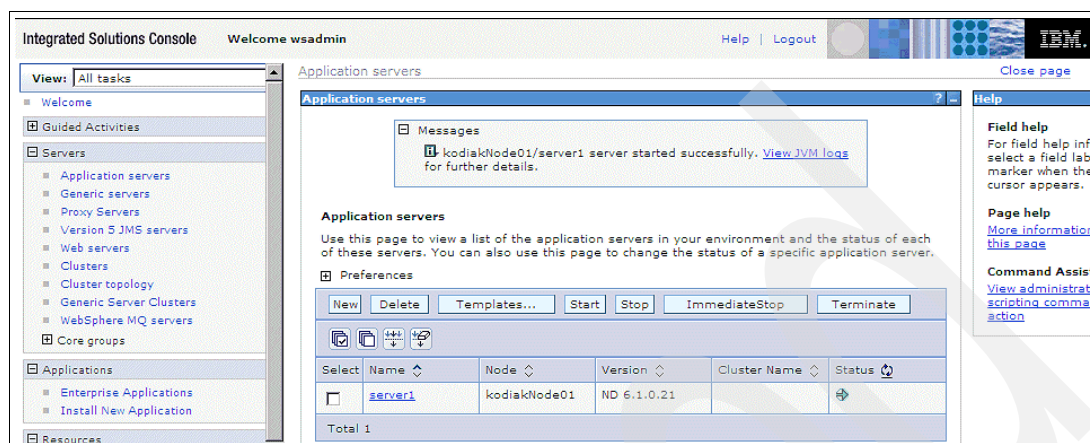


Figure B-9 Restart application server from WebSphere Application Server admin console

Your trade application should be accessible from your browser. You need to access your installation of the trade application to finish your installation.

Enter the following URL using your browser. The panel shown in Figure B-10 will display `http://<hostname>:9080/trade`

Navigate to **Configuration** → **(Re)-populate Trade Database** to finish your installation. This will populate your trade database with fictitious users and stocks. This step will take some time. You can close your browser but you will not see the status when you do so.

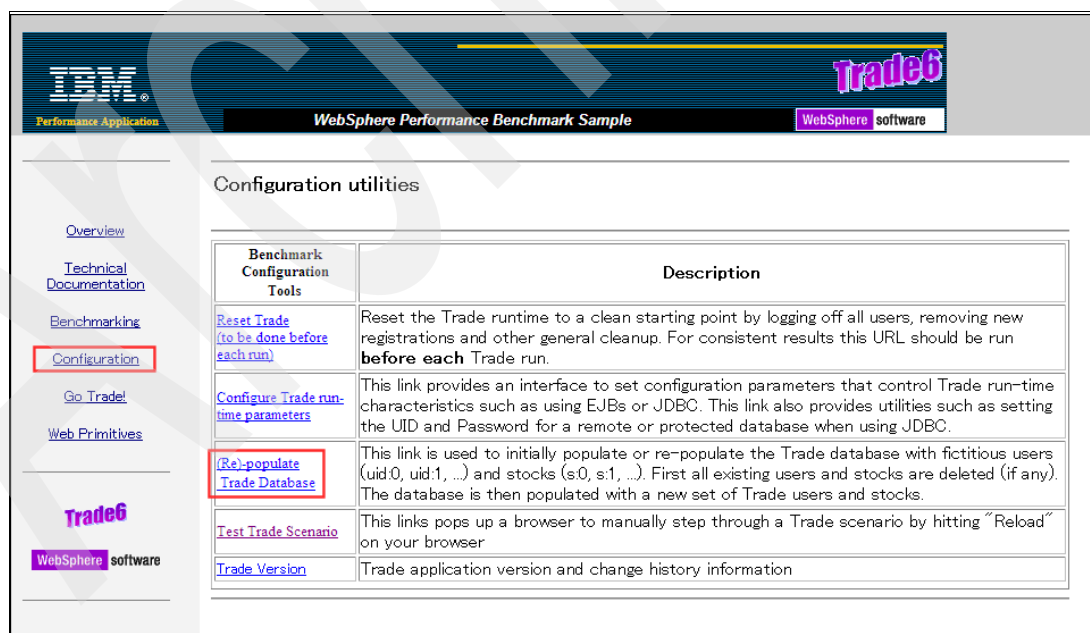


Figure B-10 Finish installation by populating Trade Database



**Tip:** If your application does not work, you may need your WebSphere Application Server administrator for help.

To work through issues on your own, we recommend reading *Approach to Problem Determination in WebSphere Application Server V6*, REDP-4073, available at the following Web page:

<http://www.redbooks.ibm.com/abstracts/redp4073.html?Open>

Also, refer to *WebSphere Application Server V6: Default Messaging Provider Problem Determination*, REDP-4076, at the following Web page:

<http://www.redbooks.ibm.com/abstracts/redp4076.html?Open>

## B.3 Using Trade

In this section, we briefly explain how the Trade application works.

Click **Go Trade!** from left menu. A panel like Figure B-11 will display. Click **Log in** where Username and Password are already inputted to get started, or create a new account by clicking **Register With Trade**.

The screenshot shows the Trade application interface. At the top, there's a blue banner with the IBM logo on the left and 'Trade6' on the right. Below the banner, a navigation menu on the left lists: Overview, Technical Documentation, Benchmarking, Configuration, Go Trade! (which is highlighted with a red border), and Web Primitives. The main content area is titled 'Trade Login' and contains a 'Log in' section. This section has two input fields: 'Username' with the value 'uid:0' and 'Password' which is masked with three dots. To the right of these fields is a 'Log in' button. Below the login fields, there's a section for new users: 'First time user? Please Register' with a 'Register With Trade' link. At the bottom of the page, there's a footer that reads: 'Created with IBM WebSphere Application Server and WebSphere Studio Application Developer Copyright 2000, IBM Corporation'.

Figure B-11 Verify your installation by logging into Trade

After you log in to Trade, you will see the “Trade Home” panel (Figure B-12). Navigate to **Home** → **Account** → **Portfolio**, or to **Quotes/Trade** to start trading.

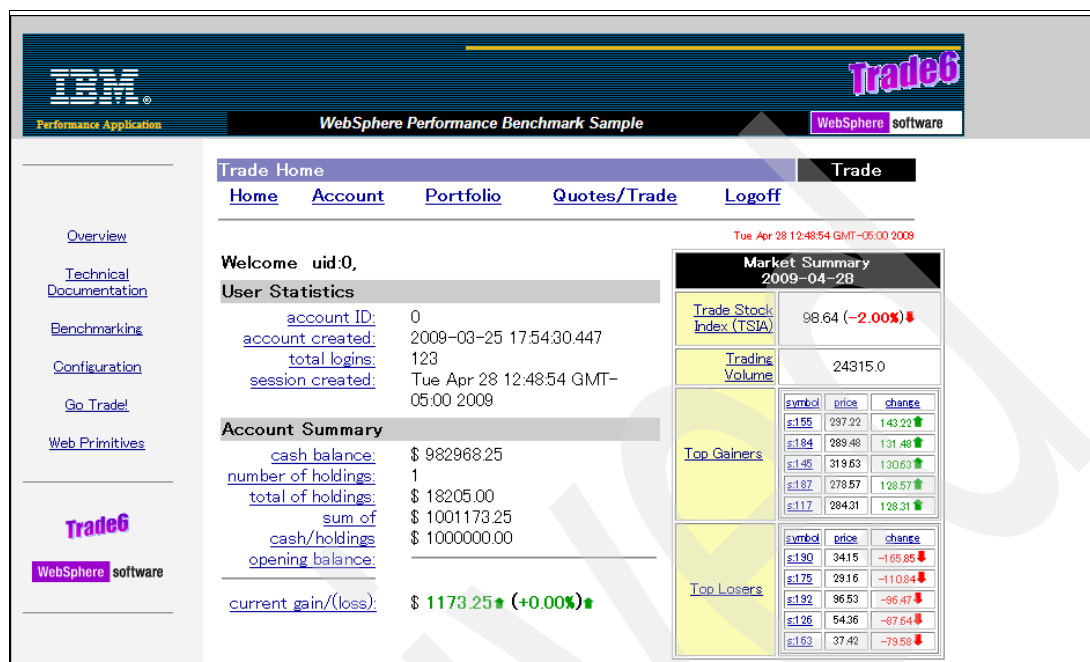


Figure B-12 Trade home panel

After you verify your installation by going through all your menu, you can start your workload using Test Trade Scenario.

Go to **Configuration** from your left menu and click **Test Trade Scenario** as shown in Figure B-13, which will display a new browser. Click **Reload**.

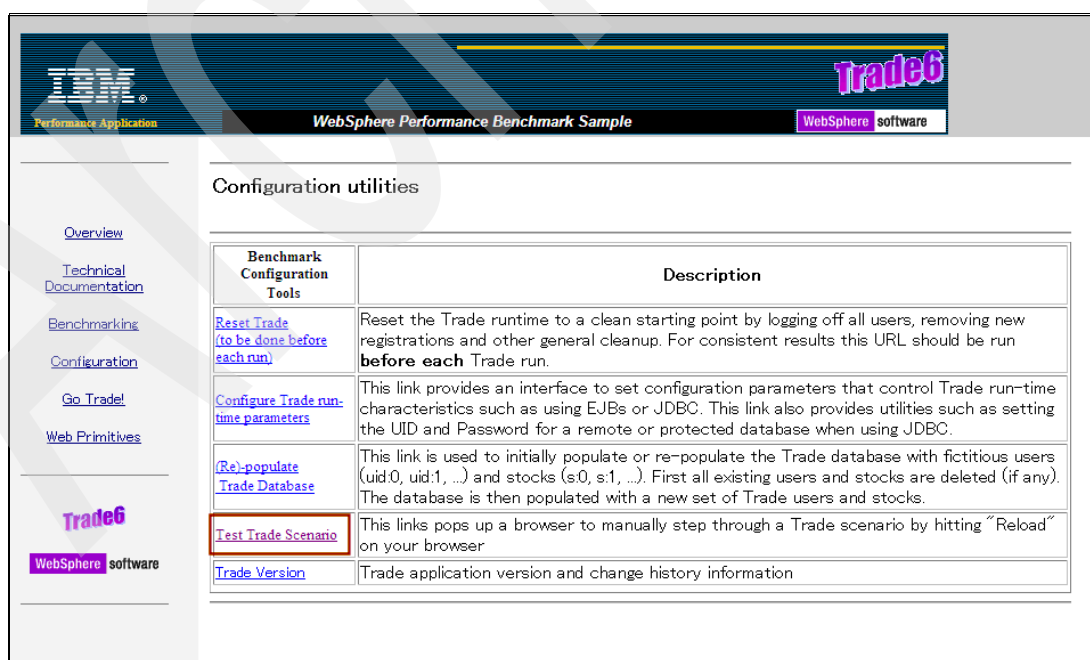


Figure B-13 Test Trade scenario

You can use any kind of load testing tool to create a workload to execute by clicking **Test Trade Scenario**. For our test, we used Apache JMeter. This is a Java-based application designed to test functional behavior and measure performance. It is available from the following Web page:

<http://jakarta.apache.org/>

Apache JMeter can load and performance test various server types but, because the Trade application provides the **Test Trade Scenario**, you can use it to test a scenario using HTTP request.

## Sample applications

This appendix provides details on the sample programs used when setting up the test cases for Chapter 5, “Application programming” on page 185.

The samples are grouped as follows:

- ▶ Appendix C.1, “Sample Java program to call a remote native SQL procedure” on page 420.
- ▶ Appendix C.2, “XA transaction samples” on page 423.
- ▶ Appendix C.3, “Progressive streaming” on page 439.

## C.1 Sample Java program to call a remote native SQL procedure

This appendix lists a Java sample program for using a native SQL procedure. This program is utilized in 5.7.10, "Remote external stored procedures and native SQL procedures" on page 224.

### C.1.1 Using the Type 4 driver to call a native SQL procedure (BSQLAlone.Java)

Example C-1 shows a Java program that creates a native SQL procedure on the DB2 9 for z/OS server, calls the procedure, opens a result set cursor and fetches from the open cursor.

#### *Example C-1 SQL procedure CALL*

---

```
// Java program that creates a native SQL procedure, calls the procedure,
// opens a result set cursor and fetches from it.

import java.io.CharArrayReader;
import java.io.Reader;
import java.sql.*;
import java.io.*;

public class BSQL_alone {

    public static Connection con = null;

    public static CallableStatement cstmt;

    public static ResultSet results;

    public static boolean debug = true;

    public static void main(String args[]) throws Exception {

        // Edit properties to match your server
        String url =
            "jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A:traceFile=BSQLTrace.txt;traceLevel=" +
            com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL + ";";

        String userid = "PAOLOR3";
        String pwd = "xxxx";

        try {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
        } catch (java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        con = DriverManager.getConnection(url, userid, pwd);

        con.setAutoCommit(false);

        createDDL(con);
        call1(con);
    }
}
```



```

        dropDDL(con);
    }

    public static void call1(Connection con) {
        String spmsg = "incoming msg";
        try {
            cstmt = con
                .prepareCall("{CALL PAOLOR3.BSQL_ALONE(?,?,?,?,?,?)}")

            byte[] inputByteArray1 = { 1,1,1};
            cstmt.setBytes(1, inputByteArray1);

            byte[] inputByteArray2 = { 2,2,2};
            cstmt.setBytes(2, inputByteArray2);

            byte[] inputByteArray3 = { 3,3,3};
            cstmt.setBytes(3, inputByteArray3);

            byte[] inputByteArray4 = { 4,4,4};
            cstmt.setBytes(4, inputByteArray4);

            cstmt.registerOutParameter(3, Types.BINARY); // IO
            cstmt.registerOutParameter(4, Types.VARBINARY);
            cstmt.registerOutParameter(5, Types.BINARY); // out as null
            cstmt.registerOutParameter(6, Types.VARBINARY); // out as null

            cstmt.execute();

            // Get the output from the inout and out params

            byte[] out_p5 = cstmt.getBytes(3);
            String sOut = "";
            for (int i = 0; i < out_p5.length; i++) {
                sOut = sOut + out_p5[i];
            }
            if (debug)
                System.out.println("out binary returned value = " + sOut);

            byte[] out_p6 = cstmt.getBytes(4);
            String sOut2 = "";
            for (int i = 0; i < out_p6.length; i++) {
                sOut2 = sOut2 + out_p6[i];
            }
            if (debug)
                System.out.println("out varbinary returned value = " + sOut2);

            out_p6 = cstmt.getBytes(5);
            sOut = "";
            for (int i = 0; i < out_p6.length; i++) {
                sOut = sOut + out_p6[i];
            }
            if (debug)
                System.out.println("in as out binary returned value = " + sOut);

            out_p6 = cstmt.getBytes(6);
            sOut2 = "";
            for (int i = 0; i < out_p6.length; i++) {
                sOut2 = sOut2 + out_p6[i];
            }
        }
    }

```

```

        if (debug)
            System.out.println("in as out varbinary returned value = "
                               + sOut2);

        System.out.println("1st CALL to SPBIN OK");

        cstmt.close();

        // Get the results of the data INPUT into tables by the in Param
    } catch (SQLException e) {
        System.out.println("*** SQLException in SECOND CALL SPBIN!" + e);
        System.out.println("****SQLCODE = " + e.getErrorCode());
        System.out.println("****SQLSTATE = " + e.getSQLState());
        System.out.println("****Text of Error Message = " + e.getMessage());
    }

    // end of try - 2nd call
}

public static void createDDL(java.sql.Connection con)
    throws java.sql.SQLException {
    try{
        Statement stmt = con.createStatement();

        // DROP is commented out the first time.
        // If you are going to run this again, uncomment the DROP.
        // stmt.executeUpdate("DROP PROCEDURE BSQL_ALONE;");
        // con.commit();

        System.out.println("DROP DONE");

        stmt.executeUpdate("CREATE TABLE PAOLOR3.CDS_1 ("
            + "Column1 CHAR(5) NOT NULL,"
            + "Column2 DECFLOAT(34),"
            + "Column3 DECFLOAT(16),"
            + "Column4 BINARY(5),"
            + "Column5 VARBINARY(40)"
            + ");");
        con.commit();

        System.out.println("CREATE TABLE DONE");

        stmt.executeUpdate("insert into PAOLOR3.CDS_1 values " +
            "('test', 12, 34, cast(x'2020' as binary(5)), " +
            "cast(x'aa123456ef' as varbinary(40)))");
        con.commit();

        System.out.println("INSERT DONE");

        stmt.executeUpdate("CREATE PROCEDURE BSQL_ALONE ( IN VAR01 BINARY(5),"
            + "          IN VAR02 VARBINARY(5),"
            + "          INOUT VAR03 BINARY(5),"
            + "          INOUT VAR04 VARBINARY(5),"
            + "          OUT VAR05 BINARY(5),"
            + "          OUT VAR06 VARBINARY(5) )"
            + "VERSION VERSION1 "
            + "ISOLATION LEVEL CS "
            + "RESULT SETS 1 "
            + "LANGUAGE SQL ");
    }
}

```

```

+ " P1: BEGIN "
+ "   DECLARE cursor1 CURSOR FOR "
+ "   \"SELECT COLUMN4, COLUMN5 \"
+ "   \" FROM CDS_1 \"
+ "   \" WHERE COLUMN5 LIKE VAR03 AND COLUMN4 NOT LIKE VAR04; \"
+ "   SET VAR05 = VAR03; \"
+ "   SET VAR06 = VAR04; \"
+ "   OPEN cursor1; \"
+ "   FETCH cursor1 INTO VAR03, VAR04; \"
+ "   END P1\";

con.commit();
System.out.println(" created st proc - Done");
stmt.close();
} catch (SQLException e) {
    System.out.println("ERROR");
    System.out.println("****SQLCODE = " + e.getErrorCode());
    System.out.println("****SQLSTATE = " + e.getSQLState());
    System.out.println("****Text of Error Message = " + e.getMessage());
}
}

public static void dropDDL(java.sql.Connection con)
    throws java.sql.SQLException {

    try {
        java.sql.Statement stmt = con.createStatement();
        stmt.executeUpdate("DROP PROCEDURE PAOLOR3.BSQL_ALONE RESTRICT");
        con.commit();
    } catch (SQLException e) {
    }

}

}

```

---

## C.2 XA transaction samples

The following two samples were used for the tests described in 5.8, “XA transactions” on page 225.

- ▶ Creating and Registering an XA datasource createRegisterXADS.java
  - createRegisterXADS.java
- ▶ Performing a 2PC update using XA transactions (XAtest.java). We have added the XAUtil class and our sample properties file in case you are hard coding the XA program.
  - Test application for XA transaction (XATest.java)

### C.2.1 createRegisterXADS.java

Example C-2 on page 424 Creates an XA DataSource and registers in JNDI (using a File System based lookup)

### Example C-2 Creating and registering an XA datasource

---

```
// Demonstrates the use of XA transactions, Section 5.8
// Should be compiled before running XAtest.java and uses the properties
// file XA_T4S390_DS2.prop
// This application creates and registers the XA datasource
// May 13, 2009

import java.sql.*;
import com.ibm.db2.jcc.DB2SimpleDataSource;
import com.ibm.db2.jcc.DB2XADataSource;
import javax.naming.*;grte
import java.util.*;
import java.io.*;

/* *****
 * Create & Register DataSources as specified by the XADS*.prop files.
 * The DataSources will be created in the temp directory.
 * It uses File System SPI - Creates a .bindings file in
 * temp directory C:/temp (as defined in jndi.properties)
 *
 * *****
 */

public class createRegisterXADS
{
    public static void main (String args[])
    {
        if (args.length != 1)
        {
            System.out.println( "Usage: createRegisterDS <name of Datasource Property File>
");
            return;
        }
        createRegisterXADS crds = new createRegisterXADS();
        System.out.println ("");
        crds.runThis(args[0]);
        System.out.println(">>>Registering XA DataSource using Property File: " + args[0] );
        System.out.println(">>>Done Register");
        System.out.println ("");
    }
    private void runThis(String args)
    {
        try
        {
            registerDS( args, new InitialContext());
        }
        catch (Exception e)
        {
            System.err.println ("Problem with registration: " + e.getMessage());
            e.printStackTrace();
            return;
        }
    }
    private void registerDS( String DSname, Context registry)
        throws Exception
    {
        DB2XADataSource dataSource = new DB2XADataSource();
    }
}
```

```

Properties prop = new Properties();
FileInputStream dsPropFile = null;
try
{
    dsPropFile = new FileInputStream( DSname);
}
catch (FileNotFoundException fe)
{
    System.out.println (fe.getMessage());
    throw fe;
}
prop.load( dsPropFile );
dataSource.setDatabaseName (prop.getProperty("databaseName"));
dataSource.setUser (prop.getProperty("userName"));
dataSource.setPassword (prop.getProperty("password"));

String DSName = prop.getProperty("dataSourceName");
dataSource.setDataSourceName (DSName);

String svName = prop.getProperty("serverName");
if (svName != null)
{
    dataSource.setServerName (svName);
}

String portNum = prop.getProperty("portNumber");
if (portNum != null)
{
    int portNo = (new Integer(portNum)).intValue() ;
    dataSource.setPortNumber (portNo);
}

String dType = prop.getProperty("driverType");
if (dType != null)
{
    int drType = (new Integer(dType)).intValue() ;
    dataSource.setDriverType(drType);
}

String sMech = prop.getProperty("securityMechanism");
if (sMech != null)
{
    short secMech = (new Integer(sMech)).shortValue();
    dataSource.setSecurityMechanism (secMech);
}

String kbp = prop.getProperty("kerberosPrincipal");
if (kbp != null)
{
    dataSource.setKerberosServerPrincipal (kbp);
}

String traceFile = prop.getProperty("traceFile");
if (traceFile != null)
{
    dataSource.setTraceFile(traceFile);
}

String traceDirectory = prop.getProperty("traceDirectory");
if (traceDirectory != null)

```

```

        {
            dataSource.setTraceDirectory(traceDirectory);
        }

        String sqlid = prop.getProperty("currentSQLID");
        if (sqlid != null)
        {
            dataSource.setCurrentSQLID(sqlid);
        }

        String traceFileAppend = prop.getProperty("traceFileAppend");
        if (traceFileAppend != null)
        {
            boolean btraceFileAppend =
                (traceFileAppend.equalsIgnoreCase ("true") ||
                 traceFileAppend.equalsIgnoreCase ("yes"));
            dataSource.setTraceFileAppend(btraceFileAppend);
        }

        String tLevel = prop.getProperty("traceLevel");
        if (tLevel != null)
        {
            int traceLevel = -1;
            if ( !tLevel.equalsIgnoreCase("0xFFFFFFFF") )
            {
                traceLevel = Integer.parseInt( tLevel.substring(2),16);
            }
            dataSource.setTraceLevel(traceLevel);
        }

        registry.rebind(DSName, dataSource);
    }
}

```

---

## C.2.2 Test application for XA transaction (XATest.Java)

Example C-3 shows a sample JDBC application running an XA transaction using the JDBC XA API. It inserts a row into a table on the DB2 for z/OS server using DRDA two-phase COMMIT flows.

*Example C-3 Sample JDBC application running an XA transaction using the JDBC XA API*

---

```

// Demonstrates the use of XA Transactions, Section 5.8
// First compile createRegisterXADS.java that uses a datasource properties file
// provided in XA_T4S390.prop
// Then compile and execute XATest.java

// This application inserts into a table that should exist in your DB2 for z/OS server
// using DRDA two-phase commit flows.
// Here is the DDL for the table:
// SET CURRENT SCHEMA='MYSCHEMA';
// DROP TABLE H_POSUPDATE;
// COMMIT;
// CREATE TABLE H_POSUPDATE (INTEGERCOL1 INT, INTEGERCOL2 INT);
// COMMIT;
// INSERT INTO H_POSUPDATE VALUES (100, 100);
// INSERT INTO H_POSUPDATE VALUES (101, 101);
// INSERT INTO H_POSUPDATE VALUES (102, 102);
// INSERT INTO H_POSUPDATE VALUES (103, 103);

```

```

// COMMIT;

import java.sql.*;
import javax.sql.*;

import javax.transaction.xa.*;
import javax.naming.*;
import java.util.*;
import java.io.*;

/*
// Test Description here:
Create Indoubt Transaction
*/

class XATest
{
    XADataSource xaDS;
    XAConnection xaconn;
    XAConnection xaconn2;
    XAResource xares;
    Xid xid;
    Connection conn;

    public static void main (String args [])
        throws SQLException
    {
        XATest xat = new XATest();
        xat.runThis(args);
    }
    public void runThis(String[] args)
    {
        // Get The Singleton XAUtil Object
        XAUtil xau = XAUtil.getXAUtil();
        String[] Args = xau.getArgs(args);
        String serverType = null;
        String driverType = null;
        serverType = Args[0];
        driverType = Args[1];

        xau.loadDriver(driverType);
        int rc = 0;
        int i = 0;

        try
        {
            InitialContext context = new javax.naming.InitialContext();
            xaDS = (XADataSource)context.lookup (xau.getDataSourceName(serverType,
driverType,1));

            xaconn = xau.getXAConnection(xaDS);
            System.out.println ("XA Connection Obtained Successfully.....");

            conn = xaconn.getConnection();
            System.out.println ("Underlying Physical Connection conn: " +
conn.getClass().getName());

```

```

// Get the XA Resources
xares = xaconn.getXAResource();

Xid xid = xau.createRandomXid();
Statement stmt1 = conn.createStatement();

int count1 = stmt1.executeUpdate("SET CURRENT SCHEMA = 'MYSCHEMA'");
xares.start (xid, XAResource.TMNOFLAGS);
xau.addToXidList(xid);
count1 = stmt1.executeUpdate("INSERT INTO H_POSUPDATE VALUES(330,340)");
xares.end(xid, XAResource.TMSUCCESS);

System.out.println( "call xares.rollback(xid)" );
xares.rollback(xid);

System.out.println( "Closing XA connection" );
xaconn.close();
xaconn = null;
}
catch (SQLException sqe)
{
    System.out.println(" SQL Exception Caught: " + sqe.getMessage());
    sqe.printStackTrace();
}
catch (XAException xae)
{
    System.out.println("XA Error is " + xae.getMessage());
    xae.printStackTrace();
}
catch (NamingException nme)
{
    System.out.println(" Naming Exception: " + nme.getMessage());
}
}
}

```

---

Example C-4 lists the XAUtil class used by the Java program in Example C-3.

#### *Example C-4 XAUtil class*

---

```

import com.ibm.db2.jcc.*;
//import COM.ibm.db2.jdbc.*;

import java.sql.*;
import javax.sql.*;

import javax.transaction.xa.*;
import javax.naming.*;
import java.util.*;
import java.io.*;
import java.math.BigDecimal;

public class XAUtil
{
    static private XAUtil __XAUtil = null;
    private Vector xidList = null;

    DB2XADataSource XADS;
    XAConnection XAConn;

```



```

XAResource XARes;

static final int XA_COMMIT=1;
static final int XA_ROLLBACK=2;
static String userName = null;
static String passwd = null;

// Constructor private only one instance can exist
private XAUtil()
{
}

static public XAUtil getXAUtil()
{
    if (__XAUtil == null)
    {
        __XAUtil = new XAUtil();
    }
    return __XAUtil;
}

/**
 * toHexString(byte[] buf)
 * Converts a byte array to a hexadecimal (base 16) character string representation.
 * @param byteBuf - a byte array
 * @return the string representation of the byte[] parameter in hexadecimal (base 16).
 */
public static String toHexString(byte[] byteBuf)
{
    char[] digits = {
        '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9', 'a', 'b',
        'c', 'd', 'e', 'f' };

    // each nibble becomes a char and there are two nibbles per byte
    char[] charBuf = new char[byteBuf.length * 2];
    int charPos = charBuf.length;
    int mask = 15; // makes the low order byte 00001111
    byte b;
    while (charPos > 0)
    {
        // Could have done an inner loop here, but only making the
        // assignment into buf twice and the shift once.
        charPos--;
        b = byteBuf[charPos/2];
        charBuf[charPos] = digits[b & mask];
        b >>= 4;
        charPos--;
        charBuf[charPos] = digits[b & mask];
    }

    return new String(charBuf, 0, charBuf.length);
}

private void getXAResFromXADS(String XADDataSource) throws NamingException,SQLException
{
    InitialContext context = new javax.naming.InitialContext();
    XADS = (DB2XADDataSource)context.lookup (XADDataSource);
    XAConn = getXAConnection(XADS);
    XARes = XAConn.getXAResource();
}

synchronized public static XAConnection getXAConnection(XADDataSource xaDS) throws SQLException
{
    String clsName = xaDS.getClass().getName();

```

```

        //if ( clsName.equals("COM.ibm.db2.jdbc.DB2XADataSource") )
        // return ((COM.ibm.db2.jdbc.DB2XADataSource)xaDS).getXAConnection (userName,passwd);
        //else
        return xaDS.getXAConnection ();
    }

    public static void loadDriver(String drType)
    {
        String driverJCC = "com.ibm.db2.jcc.DB2Driver";
        String oldt2Driver = "COM.ibm.db2.jdbc.app.DB2Driver";
        if ( drType.equals("T4") || drType.equals("T2"))
        {
            try
            {
                // register the driver with DriverManager
                Class.forName(driverJCC);
                System.out.println(" Loaded JCC(T4/T2) Driver...");
            }
            catch (ClassNotFoundException e)
            {
                System.out.println ("**** Failed to load JCC JDBC Driver ****");
                e.printStackTrace();
            }
        }
        else if (drType.equals("OLDT2") )
        {
            try
            {
                // register the driver with DriverManager
                Class.forName(oldt2Driver);
                System.out.println(" Loaded OLD T2 Driver...");
            }
            catch (ClassNotFoundException e)
            {
                System.out.println ("**** Failed to load OLDT2 JDBC App Driver ****");
                e.printStackTrace();
            }
        }
        else
        {
            System.out.println(" Wrong driverType...");
            System.exit(-1);
        }
    }

    // Check Platform 390 or UWO DriverType T2/T4/OLDT2??
    public static String[] getArgs(String[] args)
    {
        // Parse command line option.
        String platform_arg = null;
        String driverType_arg = null;
        String[] argsArray = null;
        int index = 0;
        if (args.length != 4)
        {
            System.out.println("Wrong no of Args (Enter Server and Driver Type): -s <s390|uwo> -d
            <t2|t4|oldt2> ");
            System.out.println("");
            System.exit(-1);
        }
        while (index < args.length )
        {
            if(args[index].equalsIgnoreCase("-s"))
            {
                try
            
```

```

        {
            index ++;
            if(args[index].equalsIgnoreCase("s390"))
                platform_arg = "S390";
            else if(args[index].equalsIgnoreCase("uwo"))
                platform_arg = "UWO";
            else
            {
                System.out.println(" Error: Specify Correct Server Option  -s <s390|uwo>");
                System.exit(-1);
            }
        }
        catch (Exception ex1)
        {
            System.out.println(" Exception Caught in Processing Args: " + ex1.getMessage());
            System.out.println("");
            System.exit(-1);
        }
    }
    else if(args[index].equalsIgnoreCase("-d"))
    {
        try
        {
            index ++;
            if(args[index].equalsIgnoreCase("t2"))
                driverType_arg = "T2";
            else if(args[index].equalsIgnoreCase("oldt2"))
                driverType_arg = "OLDT2";
            else if(args[index].equalsIgnoreCase("t4"))
                driverType_arg = "T4";
            else
            {
                System.out.println(" Error: Specify Correct Driver Option  -d <t2|t4|oldt2>");
                System.exit(-1);
            }
        }
        catch (Exception ex1)
        {
            System.out.println(" Exception Caught in Processing Args: " + ex1.getMessage());
            System.out.println("");
            System.exit(-1);
        }
    }
    else
    {
        System.out.println(" Unrecognized Option: Specify  -s <s390|uwo>  -d <t2|t4|oldt2>");
        System.out.println("");
        System.exit(-1);
    }
    index++;
}

argsArray = new String[2];
argsArray[0] = platform_arg;
argsArray[1] = driverType_arg;
return argsArray;
}

public static String getDataSourceName(String platformStr, String dtypeStr, int dsNo)
throws SQLException
{
    String propFileName = "XA_"+dtypeStr+platformStr+"_DS"+dsNo+".prop";
    FileInputStream dsPropFile = null;
    Properties prop = new Properties();
    try
    {

```

```

        dsPropFile = new FileInputStream( propFileName);
        prop.load( dsPropFile );
    }
    catch (FileNotFoundException fe)
    {
        System.out.println (fe.getMessage());
        throw new SQLException(fe.getMessage());
    }
    catch (IOException e)
    {
        System.out.println (e.getMessage());
        throw new SQLException(e.getMessage());
    }
    userName = prop.getProperty("userName");
    passwd = prop.getProperty("password");
    try
    {
        dsPropFile.close();
    }
    catch (IOException e)
    {
        System.out.println (e.getMessage());
        throw new SQLException(e.getMessage());
    }
    return ( "jdbc/XADataSource" + dsNo + "_" + dtypeStr + platformStr ) ;
}

public static Connection getNonXAConnection(String platformStr, String dtypeStr, int dsNo)
throws SQLException
{
    Connection conn = null;
    String propFileName = "XA_"+dtypeStr+platformStr+"_DS"+dsNo+".prop";
    FileInputStream dsPropFile = null;
    Properties prop = new Properties();
    try
    {
        dsPropFile = new FileInputStream( propFileName);
        prop.load( dsPropFile );
    }
    catch (FileNotFoundException fe)
    {
        System.out.println (fe.getMessage());
        throw new SQLException(fe.getMessage());
    }
    catch (IOException e)
    {
        System.out.println (e.getMessage());
        throw new SQLException(e.getMessage());
    }
    String ret_Str[] = new String[6];
    ret_Str[0] = prop.getProperty("driverType");
    ret_Str[1] = prop.getProperty("serverName");
    ret_Str[2] = prop.getProperty("portNumber");
    ret_Str[3] = prop.getProperty("databaseName");
    ret_Str[4] = prop.getProperty("userName");
    ret_Str[5] = prop.getProperty("password");

    try
    {
        dsPropFile.close();
    }
    catch (IOException e)
    {
        System.out.println (e.getMessage());
        throw new SQLException(e.getMessage());
    }
}

```

```

    }
    String URL=null;
    if (ret_Str[0].equals("2"))
    {
        URL = "jdbc:db2:"+ret_Str[3];
    }
    else if (ret_Str[0].equals("4"))
    {
        URL = "jdbc:db2://" + ret_Str[1] + ":" + ret_Str[2] + "/" + ret_Str[3] ;
    }
    Properties p = new Properties();
    p.put("user", ret_Str[4]);
    p.put("password", ret_Str[5]);
    System.out.println(" Obtain Non XA Connection .... " );
    conn = DriverManager.getConnection(URL,p);

    return ( conn ) ;
}

// Remove an Xid from XidList
public boolean removeFromXidList(Xid xid)
{
    boolean retStatus = false;

    if (xidList != null && xidList.size() > 0)
        retStatus = xidList.removeElement(xid);

    return retStatus;
}

/* This enters an xid to the maintained xidListVector */
public void addToXidList(Xid xid)
{
    if (xidList == null )
    {
        xidList = new Vector(20);
    }
    if ( (xid != null) && !(xidList.contains(xid)) )
    {
        xidList.addElement(xid);
    }
}

/* This removes xidList Vector */
public void removeXidList()
{
    if (xidList != null)
    {
        xidList.removeAllElements();
        xidList = null;
    }
}

public Xid findInXidList(Xid xid )
{
    if (xidList != null && xidList.contains(xid) )
        return xid;
    else
        return null;
}

public int noOfElementsInxidList()
{
    if (xidList != null)

```

```

        return (xidList.size());
    return 0;
}

public void printXidList()
{
    if (xidList != null && xidList.size() > 0)
    {
        for (Enumeration e = xidList.elements(); e.hasMoreElements(); )
            System.out.println(e.nextElement());
    }
}

public Xid[] recoverTransactionForXAResource(String XADS ) throws SQLException, XAException
{
    Xid x[] = null;
    try
    {
        getXAResFromXADS(XADS);
        x = XARes.recover(XAResource.TMSTARTSCAN|XAResource.TMENDRSCAN);
        System.out.println( "Xid[] Recovered is : " + x);
    }
    catch (XAException xe)
    {
        throw new XAException ( "Recover Failed: " + xe.toString());
    }
    catch (Exception e)
    {
        throw new SQLException (e.toString());
    }
    XAConn.close();
    return x;
}

public Xid[] recoverTransactionForXAResource(String XADS, int action ) throws SQLException,
XAException
{
    Xid x[] = null;
    try
    {
        getXAResFromXADS(XADS);
        x = XARes.recover(XAResource.TMSTARTSCAN|XAResource.TMENDRSCAN);
        System.out.println( "Xid[] Recovered is : " + x);
    }
    catch (XAException xe)
    {
        throw new XAException (xe.toString());
    }
    catch (Exception e)
    {
        throw new SQLException (e.toString());
    }
    if ( (x != null) && (x.length > 0) )
    {
        for (int i=0; i < x.length; ++i)
        {
            System.out.println( "Recovered Xid: " + x[i]);
            try
            {
                if (action == XA_ROLLBACK )
                    XARes.rollback(x[i]);
                else if (action == XA_COMMIT )
                    XARes.commit(x[i],false);
            }
            else
                throw new SQLException ( "Wrong action..." );
        }
    }
}

```

```

    }
    catch (XAException xe)
    {
        try
        {
            XARes.forget(x[i]);
        }
        catch (XAException xae)
        {
            throw new XAException( "Rollback/Forget Failed: " + xae.toString());
        }
    }
}
}
XAConn.close();
return x;
}
synchronized public Xid createXid(int fid, int gids, int bids) throws XAException
{
    byte[] gid = new byte[1]; gid[0]= (byte) gids;
    byte[] bid = new byte[1]; bid[0]= (byte) bids;
    byte[] gtrid = new byte[64];
    byte[] bqual = new byte[64];
    System.arraycopy (gid, 0, gtrid, 0, 1);
    System.arraycopy (bid, 0, bqual, 0, 1);
    Xid xid = new DB2Xid(fid, gtrid, bqual);
    return xid;
}
synchronized public Xid createXid(int bids) throws XAException
{
    byte[] gid = new byte[1]; gid[0]= (byte) 9;
    byte[] bid = new byte[1]; bid[0]= (byte) bids;
    byte[] gtrid = new byte[64];
    byte[] bqual = new byte[64];
    System.arraycopy (gid, 0, gtrid, 0, 1);
    System.arraycopy (bid, 0, bqual, 0, 1);
    Xid xid = new DB2Xid(0x1234, gtrid, bqual);
    return xid;
}
synchronized public Xid createXid(int fid, byte[] gid, byte[] bqual) throws XAException
{
    Xid xid = new DB2Xid(fid, gid, bqual);
    return xid;
}

static Random rnd = new Random();
// Create a unique random xid
synchronized public Xid createRandomXid() throws XAException
{
    int fid;
    fid = rnd.nextInt();
    int len1 = Math.abs(rnd.nextInt()) % 64;
    int len2 = Math.abs(rnd.nextInt()) % 64;
    byte[] gid = new byte[len1];
    byte[] bqual = new byte[len2];
    rnd.nextBytes(gid);
    rnd.nextBytes(bqual);
    Xid xid = new DB2Xid(fid, gid, bqual);
    return xid;
}
public void dispResultSet( ResultSet rs ) throws SQLException
{
    int i ;
    int ncols ;
    ResultSetMetaData rsmd ;
    String s = null ;

```

```

String          comma = " " ;
byte []         blob ;
int             typ ;
int             blen ;

rsmd = rs.getMetaData() ;
ncols = rsmd.getColumnCount() ;

System.out.println( "number of columns " + ncols ) ;

for( i=1; i<=ncols; ++i )
{
    if( i > 1 )
    {
        if ( i < ncols)
            comma = "," ;
        else
            comma = " " ;
    }
    System.out.println((rsmd.getColumnLabel( i )).trim() + comma ) ;
}
System.out.println( "\n" ) ;

while( rs.next() )
{
    for( i=1; i<=ncols; ++i )
    {

        typ = rsmd.getColumnType( i ) ;
        //System.out.print( "typ " + typ + " " ) ;

        switch( typ ) {
            case Types.CHAR:
            case Types.VARCHAR:
            case Types.LONGVARCHAR:
                s = rs.getString( i ) ;
                break ;

            case Types.SMALLINT:
                short v = rs.getShort( i ) ;
                if( !rs.isNull() ) s = "" + v ;
                break ;

            case Types.INTEGER:
                int vi = rs.getInt( i ) ;
                if( !rs.isNull() ) s = "" + vi ;
                break ;

            case Types.DOUBLE:
                double vd = rs.getDouble( i ) ;
                if( !rs.isNull() ) s = "" + vd ;
                break ;

            case Types.REAL:
                float vfr = rs.getFloat( i ) ;
                if( !rs.isNull() ) s = "" + vfr ;
                break ;

            case Types.FLOAT:
                float vf = rs.getFloat( i ) ;
                if( !rs.isNull() ) s = "" + vf ;
                break ;

            case Types.DECIMAL:
            case Types.NUMERIC:
                BigDecimal vb = rs.getBigDecimal( i ) ;

```



```

        if( !rs.isNull() ) s = vb.toString() ;
        break ;

    case Types.DATE:
        java.sql.Date vt = rs.getDate( i ) ;
        if( !rs.isNull() ) s = vt.toString() ;
        break ;

    case Types.TIME:
        java.sql.Time vti = rs.getTime( i ) ;
        if( !rs.isNull() ) s = vti.toString() ;
        break ;

    case Types.TIMESTAMP:
        java.sql.Timestamp vts = rs.getTimestamp( i ) ;
        if( !rs.isNull() ) s = vts.toString() ;
        break ;

    default:
        s = "" ;
}
if( rs.isNull() )
{
    if( i > 1 )
    {
        if ( i < ncols)
            comma = "," ;
        else
            comma = " " ;
    }
    System.out.println("<null>" + comma ) ;
}
else
{
    if ( i < ncols)
        comma = "," ;
    else
        comma = " " ;
    System.out.println(s.trim() + comma ) ;
}
}
System.out.println( "" ) ;
}
}

public void populate(Connection conn, int start, int end)
{
    int i;
    int rc;
    int colIdx;
    String s = null;
    PreparedStatement pstmt;
    int[] acctIDTable = {10001, 10002, 10003, 10004, 10005, 10006, 10007,
                        10008, 10009, 10010};
    int[] branchIDTable = {1111, 1112, 1111, 1112, 1111, 1112, 1111, 1112,
                          1111, 1112};
    int[] balanceTable = {3000, 50000, 20000, 1000, 200, 1000000, 30, 120000,
                        10000, 99990};
    String[] acctTypeTable = {"c", "s", "c", "s", "c", "s", "c", "s", "c", "s"};
    String[] firstNameTable = {"Swami", "Manish", "Limin", "Anshul", "Jyhchen", "Del",
                              "Chunyang", "Clifford", "Hong", "Bill"};
    String[] lastNameTable = {"Gounder", "Sehgal", "Yang", "Dawra", "Fang",
                              "Blevins", "Xia", "Chu", "Yie", "Birley"};
    String[] midInitTable = {"J", "P", "", "", "", "G", "", "", "", ""};
    String[] phoneTable = {"800-CALL-IBM1", "800-DONT-CALL", "", "", "",
                          "", "", "", "", ""};
}

```

```

String[] addrTable = {"2 Royal Ann Dr, Everywhere, USA",
    "3 Marine Way Blvd, Hongkong",
    "4 El-Dorado Way , Singapore",
    "5 Asia Blvd, Asia",
    "6 IBM Street, Fremont, CA. USA",
    "7 IBM Avenue, Los Gatos, CA. USA",
    "8 IBM Way, Union City, CA. USA",
    "9 IBM Parkway, ???, CA. USA",
    "10 IBM Loop, San Jose, CA. USA",
    "11 Bailey Ave , San Jose, CA. USA"};
System.out.println("INSERT INTO CHK_ACCOUNT TABLE");

try
{
    s = "insert into CHK_ACCOUNT values (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    pstmt = conn.prepareStatement(s);

    for (i = start; i <= end; i++)
    {
        colIdx = 1;
        pstmt.setInt(colIdx++, acctIDTable[i]);
        pstmt.setInt(colIdx++, branchIDTable[i]);
        pstmt.setInt(colIdx++, balanceTable[i]);
        pstmt.setString(colIdx++, acctTypeTable[i]);
        pstmt.setString(colIdx++, firstNameTable[i]);
        pstmt.setString(colIdx++, lastNameTable[i]);
        pstmt.setString(colIdx++, midInitTable[i]);
        pstmt.setString(colIdx++, phoneTable[i]);
        pstmt.setString(colIdx++, addrTable[i]);
        rc = pstmt.executeUpdate();

        if (rc != 1)
        {
            System.out.println(" populate: insert returned " + rc + ", expects 1");
        }
    }
    pstmt.close();
}
catch (SQLException e)
{
    System.out.println(" Insert failed - sqlcode : " + e.getErrorCode());
    System.out.println(" message: " + e.toString());
}

}

public void cleanup(Connection conn)
{
    int rowCount = 0;
    Statement stmt = null;

    try
    {
        stmt = conn.createStatement();
        rowCount = stmt.executeUpdate("delete from CHK_ACCOUNT");
        if (!conn.getAutoCommit())
            conn.commit();
    }
    catch (Exception e)
    {
        // don't care
    }
}

public Enumeration enumXidList()
{

```

```

        return xidList.elements();
    }
}

```

---

Example C-5 shows the properties file we used.

*Example C-5 Properties file XADataSource1\_T4S390*

---

```

userName=paolor3
password=fri13day
databaseName=DB9C
dataSourceName=jdbc/XADataSource1_T4S390
driverType=4
serverName=wtsc63.itso.ibm.com
portNumber=38320
securityMechanism=3
kerberosPrincipal=null
traceFile=abc.txt
#traceLevel=0xFFFFFFFF

```

---

## C.3 Progressive streaming

This Appendix contains a sample of progressive streaming (using dynamic data format) with XML data as provided at 5.7.8, “Progressive streaming” on page 222:

- Progressive streaming: XMLTest\_RedJava

### C.3.1 Progressive streaming: XMLTest\_RedJava

Example C-6 on page 439 shows the use of progressive streaming to retrieve XML data. Note that although we have shown the explicit setting of `progressiveStreaming` property, this would be the default when connecting to a DB2 9 for z/OS server. The input file used by this sample program is not included.

*Example C-6 Using Progressive Streaming: XMLTest\_RedJava*

---

```

// Java program that reads from an XML file (XML2MB.xml)
// It uses progressive references to retrieve the large XML
// data in chunks.
// May 13, 2009
// Progressive Streaming: Section 5.7.6

```

```

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.InputStreamReader;
import java.io.StringReader;

import com.ibm.db2.jcc.DB2BaseDataSource;

public class XMLTest2_Red
{
    public static void main (String[] args)
    {
        // Edit the properties below to match your server
        String ServerName = "wtsc63.itso.ibm.com";

```

```

int PortNumber = 12347;
String DatabaseName = "DB9A";

String url = "jdbc:db2://" + ServerName + ":" + PortNumber + "/" +
            DatabaseName+
":traceFile=c:/temp/foobar.txt;traceLevel="+ 0xFFFFFFFF + ";";
java.util.Properties properties = new java.util.Properties();
properties.put("user", "paolor3");
properties.put("password", "xxxx");
properties.put("progressiveStreaming", "1");

java.sql.Connection con = null;
try
{
    Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
}
catch ( Exception e )
{
    System.out.println("Error: failed to load Db2 jcc driver.");
}

try
{
    System.out.println("url: " + url);

    con = java.sql.DriverManager.getConnection(url, properties);

    java.sql.Statement s = con.createStatement();
    try {
        s.executeUpdate("drop table testTable1");
    }
    catch (Exception e)
    {
        //ignore
    }

    s.executeUpdate ("create table testTable1 (i2 XML)");

    java.sql.PreparedStatement ps = con.prepareStatement ("insert into testTable1
values ( ?)");

    // Edit filename below with your input file
    String fileName = "c:\\ddf\\test\\javatests\\XML2MB.xml";
    java.io.FileInputStream fis = new java.io.FileInputStream(fileName);

    ps.setAsciiStream(1, fis, (int)fis.getChannel().size());
    ps.executeUpdate();

    java.sql.PreparedStatement ps2 = con.prepareStatement ("select * from testTable1
");

    java.sql.ResultSet rs = ps2.executeQuery ();

    while(rs.next()){
        String xml = rs.getString(1);
        System.out.println(xml.length());
        System.out.println(xml.substring(0, 150));
    }
}

```

```
        System.out.println(xml.substring(xml.length()-150, xml.length()));
    }

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
}
```

---

Archived

## Sample programs for performance analysis

This appendix includes samples of code used in Part 4, “Performance and problem determination” on page 267.

This appendix contains the following:

- ▶ D.1, “Stress tests script” on page 444
- ▶ D.2, “REXX parser of GTF trace for IFCID 180” on page 446

## D.1 Stress tests script

Example D-1 shows a shell script executing a query in a remote DB2 for a number of times. You can use this program for executing tests workloads.

*Example D-1 Korn shell script: executing a DB2 query from UNIX*

---

```
#!/usr/bin/ksh
# Create workload in z/OS from an Unix client
#
# Usage --> ./workload.sh

# Definitions
# -----

# Target DB2 for z/OS alias
MFDB2="DB9A"

# Userid at Server
HOSTuser="paolor4"

# Password at Server
HOSTpasswd="123abc"

# Number of execution of test query
count=10

# The following line defines the query to be executed during the benchmark
# Adapt the 'stmt' variable to the test, i.e. I/O or CPU bound
# The following is an example extracted from the DB2 sample job
# hld.SDSNSAMP(DSNTIJPM).

stmt="
SELECT A.NAME, A.TBNAME, B.TSNAME
  FROM SYSIBM.SYSINDEXES  A,
       SYSIBM.SYSTABLES   B,
       SYSIBM.SYSINDEXPART C
 WHERE A.TBNAME      = B.NAME
       AND A.TBCREATOR = B.CREATOR
       AND A.NAME      = C.IXNAME
       AND A.CREATOR   = C.IXCREATOR
       AND B.DBNAME    = 'DSNDB06'
       AND B.CREATEDBY = 'SYSIBM'
       AND A.IBMREQD   <> 'Y'
       AND C.STORTYPE  = 'E'
       AND B.TSNAME IN( 'SYSVIEWS', 'SYSDBASE', 'SYSDBAUT', 'SYSDDF',
                        'SYSGPAUT', 'SYSGROUP', 'SYSGRTNS', 'SYSHIST',
                        'SYSJAVA',  'SYSOBJ',  'SYSPKAGE', 'SYSPLAN',
                        'SYSSEQ',   'SYSSEQ2', 'SYSSTATS', 'SYSSTR',
                        'SYSUSER', 'SPT01' )
 ORDER BY B.TSNAME, A.TBNAME, A.NAME;"

# Start program
# -----
```



```

echo "Connecting to "$MFDB2
db2 "Connect to " $MFDB2 " user " $HOSTuser " using " $HOSTpasswd
while [[ $count -gt 0 ]];do
print "\$count is $count"
(( count -= 1 ))

echo $stmt
db2 -xt $stmt
done

# End program
# -----

db2 terminate

```

Output example:

-----

Connecting to DB9A

#### Database Connection Information

```

Database server      = DB2 OS/390 9
SQL authorization ID = paolor4
Local database alias = DB9A

```

```

$count is 10
SELECT A.NAME, A.TBNAME, B.TSNAME FROM SYSIBM.SYSINDEXES A,
SYSIBM.SYSTABLES B, SYSIBM.SYSINDEXPART C WHERE A.TBNAME = B.NAME AND
A.TBCREATOR = B.CREATOR AND A.NAME = C.IXNAME AND A.CREATOR = C.IXCREATOR
AND B.DBNAME = 'DSNDB06' AND B.CREATEDBY = 'SYSIBM' AND A.IBMREQD <> 'Y'
AND C.STORTYPE = 'E' AND B.TSNAME IN( 'SYSVIEWS', 'SYSDBASE', 'SYSDBAUT',
'SYSDDF', 'SYSGPAUT', 'SYSGROUP', 'SYSGRTNS', 'SYSHIST', 'SYSJAVA' ,
'SYSOBJ' , 'SYSPKAGE', 'SYSPLAN', 'SYSSEQ' , 'SYSSEQ2' , 'SYSSTATS',
'SYSSTR', 'SYSUSER' , 'SPT01' ) ORDER BY B.TSNAME, A.TBNAME, A.NAME;
$count is 9
SELECT A.NAME, A.TBNAME, B.TSNAME FROM SYSIBM.SYSINDEXES A,
SYSIBM.SYSTABLES B, SYSIBM.SYSINDEXPART C WHERE A.TBNAME = B.NAME AND
A.TBCREATOR = B.CREATOR AND A.NAME = C.IXNAME AND A.CREATOR = C.IXCREATOR
AND B.DBNAME = 'DSNDB06' AND B.CREATEDBY = 'SYSIBM' AND A.IBMREQD <> 'Y'
AND C.STORTYPE = 'E' AND B.TSNAME IN( 'SYSVIEWS', 'SYSDBASE', 'SYSDBAUT',
'SYSDDF', 'SYSGPAUT', 'SYSGROUP', 'SYSGRTNS', 'SYSHIST', 'SYSJAVA' ,
'SYSOBJ' , 'SYSPKAGE', 'SYSPLAN', 'SYSSEQ' , 'SYSSEQ2' , 'SYSSTATS',
'SYSSTR', 'SYSUSER' , 'SPT01' ) ORDER BY B.TSNAME, A.TBNAME, A.NAME;
$count is 8
....
....
$count is 1
SELECT A.NAME, A.TBNAME, B.TSNAME FROM SYSIBM.SYSINDEXES A,
SYSIBM.SYSTABLES B, SYSIBM.SYSINDEXPART C WHERE A.TBNAME = B.NAME AND
A.TBCREATOR = B.CREATOR AND A.NAME = C.IXNAME AND A.CREATOR = C.IXCREATOR

```

```

AND B.DBNAME = 'DSNDB06' AND B.CREATEDBY = 'SYSIBM' AND A.IBMREQD <> 'Y'
AND C.STORTYPE = 'E' AND B.TSNAME IN( 'SYSVIEWS', 'SYSDBASE', 'SYSDBAUT',
'SYSDDF', 'SYSGPAUT', 'SYSGROUP', 'SYSGRTNS', 'SYSHIST', 'SYSJAVA' ,
'SYSOBJ' , 'SYSPKAGE', 'SYSPLAN', 'SYSSEQ' , 'SYSSEQ2' , 'SYSSTATS',
'SYSSTR', 'SYSUSER' , 'SPT01' ) ORDER BY B.TSNAME, A.TBNAME, A.NAME;
DB20000I The TERMINATE command completed successfully.

```

---

Example D-2 shows an example of script that will call and execute in the background the previous example.

*Example D-2 Korn shell: executing a query script in the background*

---

```

./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &

./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &
./work.sh > /dev/null &

```

---

## D.2 REXX parser of GTF trace for IFCID 180

In “Creating a trace reporting tool” on page 381 we describe how to create a REXX program that will assemble GTF records and parse the IFCID 180 in a report. This section contains the JCL and REXX code described in that section.

*Example D-3 JCL for execution of REXX parser of GTF containing IFCID 180*

---

```

/*-----
/* DRDA REDBOOK --> REXX FOR FORMATING GTF TRACES
/*-----
//RUN      EXEC PGM=IKJEFT01
//SYSPROC DD  DISP=SHR,DSN=PAOLOR4.DRDA.UTIL.CNTL
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//TRACES   DD  DISP=SHR,DSN=PAOLOR4.GTFDRDA.XXX
/* REXX NAME
/*
/*      DB2 SUBSYSTEM
/*      |      TRACE TYPE
/*      |      |      DISPLAY DATA OPTION (DATA=Y | DATA=N)
/*      V      V      V      V
/* PARSEREX DB9A  GTF DATA=Y
//SYSTSIN DD  *
PARSEREX DB9A  GTF DATA=N

```

---

Example D-4 on page 447 contains the code used for parsing a GTF trace containing IFCID 180 records.

Example D-4 REXX code: parsing of GTF trace and IFCID 180

```

/* REXX */                                00001000
/*-----*/                                00002014
/* WARNING */                              00003014
/* This program is an example and provided as is */ 00003114
/* Very limited tests were done in order to validate how it works */ 00003214
/* This program is intended to be a teaching tool */ 00003314
/* USE IT AT YOUR OWN RISK */              00003414
/*-----*/                                00004014
arg DB2 TraceType dataOption              00040011
                                           00041002

if (dataOption <> 'DATA=Y') & (dataOption <> 'DATA=N') then do 00041111
    say 'Warning: invalid option entered for display data'      00041211
    say '        Valid are DATA=Y | DATA=N'                  00041311
    say '        Parameter ignored'                             00041411
end                                                            00041511
/*-----*/                                00041714
/* definitions */                                              00041814
/*-----*/                                00041914
IFCID = 180                                                    00042011
debug_flag = "n"                                              00042117
recordcnt = 0                                                  00043006
record.0 = 0                                                   00043106
record.  = ""                                                 00044006
toprocess = 0                                                  00044110
                                           00065810
/*-----*/                                00065914
/* EBCDIC to ASCII conversion tables */                        00066014
/*-----*/                                00066114
toEBCDIC = '00010203372D2E2F1605250B0C0D0E0F'x /* 00 */ 00066210
toEBCDIC = toEBCDIC|'101112133C3D322618193F271C1D1E1F'x /* 10 */ 00066310
toEBCDIC = toEBCDIC|'405A7F7B5B6C507D4D5D5C4E6B604B61'x /* 20 */ 00067010
toEBCDIC = toEBCDIC|'F0F1F2F3F4F5F6F7F8F9A5E4C7E6E6F'x /* 30 */ 00068010
toEBCDIC = toEBCDIC|'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'x /* 40 */ 00069010
toEBCDIC = toEBCDIC|'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'x /* 50 */ 00070010
toEBCDIC = toEBCDIC|'79818283848586878889919293949596'x /* 60 */ 00080010
toEBCDIC = toEBCDIC|'979899A2A3A4A5A6A7A8A9C04FD0A107'x /* 70 */ 00090010
toEBCDIC = toEBCDIC|'202122232415061728292A2B2C090A1B'x /* 80 */ 00100010
toEBCDIC = toEBCDIC|'30311A333435360838393A3B04143EFF'x /* 90 */ 00110010
toEBCDIC = toEBCDIC|'41AA4AB19FB26AB5BBB49A8AB0CAAFBC'x /* A0 */ 00120010
toEBCDIC = toEBCDIC|'908FEAFABEA0B6B39DDA9B8BB7B8B9AB'x /* B0 */ 00130010
toEBCDIC = toEBCDIC|'6465626663679E687471727378757677'x /* C0 */ 00140010
toEBCDIC = toEBCDIC|'AC69EDEEBEFECBF80FDFEFBFCBAAE59'x /* D0 */ 00150010
toEBCDIC = toEBCDIC|'4445424643479C485451525358555657'x /* E0 */ 00160010
toEBCDIC = toEBCDIC|'8C49CDCECBCFCE170DDDEDBDC8D8EDF'x /* F0 */ 00170010
                                           00201010
toASCII = '000102039C09867F978D8E0B0C0D0E0F'x /* 00 */ 00202010
toASCII = toASCII|'101112139D8508871819928F1C1D1E1F'x /* 10 */ 00202110
toASCII = toASCII|'80818283840A171B88898A8B8C050607'x /* 20 */ 00202210
toASCII = toASCII|'909116939495960498999A9B14159E1A'x /* 30 */ 00202310
toASCII = toASCII|'20A0E2E4E0E1E3E5E7F1A22E3C282B7C'x /* 40 */ 00202410
toASCII = toASCII|'26E9EAE8E8EDEEEFECDF21242A293B5E'x /* 50 */ 00202510
toASCII = toASCII|'2D2FC2C4C0C1C3C5C7D1A62C255F3E3F'x /* 60 */ 00202610
toASCII = toASCII|'F8C9CACBC8CDCECFCC603A2340273D22'x /* 70 */ 00202710
toASCII = toASCII|'D8616263646566676869ABBBF0FDFEB1'x /* 80 */ 00202810
toASCII = toASCII|'B06A6B6C6D6E6F707172AABAE6B8C6A4'x /* 90 */ 00202910
toASCII = toASCII|'B57E737475767778797AA1BFD05BDEAE'x /* A0 */ 00203010
toASCII = toASCII|'ACA3A5B7A9A7B6BCBDBEDDA8AF5DB4D7'x /* B0 */ 00203110
toASCII = toASCII|'7B414243444546474849ADF4F6F2F3F5'x /* C0 */ 00203210
toASCII = toASCII|'7D4A4B4C4D4E4F505152B9FBFCF9FAFF'x /* D0 */ 00203310

```

```

toASCII = toASCII||'5CF7535455565758595AB2D4D6D2D3D5'x /* E0 */      00203410
toASCII = toASCII||'30313233343536373839B3DBDCD9DA9F'x /* F0 */      00203510
                                          00204010

hextable =          '000102030405060708090A0B0C0D0E0F'x /* 00 */      00204110
hextable = hextable || '101112131415161718191A1B1C1D1E1F'x /* 10 */      00204210
hextable = hextable || '202122232425262728292A2B2C2D2E2F'x /* 20 */      00204310
hextable = hextable || '303132333435363738393A3B3C3D3E3F'x /* 30 */      00204410
hextable = hextable || '404142434445464748494A4B4C4D4E4F'x /* 40 */      00204510
hextable = hextable || '505152535455565758595A5B5C5D5E5F'x /* 50 */      00204610
hextable = hextable || '606162636465666768696A6B6C6D6E6F'x /* 60 */      00204710
hextable = hextable || '707172737475767778797A7B7C7D7E7F'x /* 70 */      00204810
hextable = hextable || '808182838485868788898A8B8C8D8E8F'x /* 80 */      00204910
hextable = hextable || '909192939495969798999A9B9C9D9E9F'x /* 90 */      00205010
hextable = hextable || 'A0A1A2A3A4A5A6A7A8A9AAABACADAFAF'x /* A0 */      00205110
hextable = hextable || 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x /* B0 */      00205210
hextable = hextable || 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x /* C0 */      00205310
hextable = hextable || 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x /* D0 */      00205410
hextable = hextable || 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x /* E0 */      00205510
hextable = hextable || 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x /* F0 */      00205610
                                          00249610

/*-----*/00249714
/* DDM definition example table */      00249814
/* NO ALL THE DDM definitions are included here */      00249914
/* Refer to the following publication for more details: */      00250014
/* DRDA, Version 4, Volume 3: */      00250114
/* Distributed Data Management (DDM) Architecture - The Open Group */      00250214
/*-----*/00250614

ddmcp.1='1041';ddmde.1='EXCSAT - Exchange Server Attributes'      00250714
ddmcp.2='1443';ddmde.2='EXCSATRD - Server Attributes Reply Data'      00250814
ddmcp.3='14AC';ddmde.3='ACCSECRD - Access Security Reply Data'      00250914
ddmcp.4='106E';ddmde.4='SECCHK - Security Check'      00251014
ddmcp.5='1219';ddmde.5='SECCHKRM - Security Check Reply Message'      00251114
ddmcp.6='2201';ddmde.6='ACCRDBRM - Access RDB Reply Message'      00251214
ddmcp.7='200D';ddmde.7='PRPSQLSTT - PrepareSQL Statement'      00251314
ddmcp.8='2408';ddmde.8='SQLCARD - SQL Communications Area Reply Data'      00251414
ddmcp.9='200C';ddmde.9='OPNQRY - Open Query'      00251514
ddmcp.10='2205';ddmde.10='OPNQRYRM - Open Query Complete'      00251614
ddmcp.11='241A';ddmde.11='QRYDSC - Query Answer Set Description'      00251714
ddmcp.12='241B';ddmde.12='QRYDTA - Query Answer Set Data'      00251814
ddmcp.13='220B';ddmde.13='ENDQRYRM - End of Query'      00251914
ddmcp.14='2408';ddmde.14='SQLCARD - SQL Communications Area Reply Data'      00252014
ddmcp.15='2008';ddmde.15='DSCSQLSTT - Describe SQL Statement'      00252114
ddmcp.16='2411';ddmde.16='SQLDARD - SQLDA Reply Data'      00252214
ddmcp.17='200E';ddmde.17='RDBCMM - RDB Commit Unit of Work'      00252314
ddmcp.18='115E';ddmde.18='EXTNAM - External Name'      00252414
ddmcp.19='1404';ddmde.19='MGRLVLLS - Manager-Level List'      00252514
ddmcp.20='1147';ddmde.20='SRVCLSNM - Server Class Name'      00252614
ddmcp.21='116D';ddmde.21='SRVNAM - Server Name'      00252714
ddmcp.22='115A';ddmde.22='SRVRLSLV - Server Product Release Level'      00252814
ddmcp.23='11A1';ddmde.23='PASSWORD - Password'      00252914
ddmcp.24='11A0';ddmde.24='USRID - User ID at the Target System'      00253014
ddmcp.25='1149';ddmde.25='SVRCOD - Severity Code'      00253114
ddmcp.26='11A4';ddmde.26='SECCHKCD - Security Check Code'      00253214
ddmcp.27='11A2';ddmde.27='SECMEC - Security Mechanism'      00253314
ddmcp.28='2110';ddmde.28='RDBNAM - Relational Database Name'      00253414
ddmcp.29='002F';ddmde.29='TYPDEFNAM - Data Type Definition Name'      00253514
ddmcp.30='112E';ddmde.30='PRDID - Product-specific Identifier'      00253614
ddmcp.31='0035';ddmde.31='TYPDEF OVR - TYPDEF Overrides'      00253714
ddmcp.32='2125';ddmde.32='PKGDFTCST - Package Default Character Subtype'      00253814
ddmcp.33='244E';ddmde.33='SRVLST - Server List'      00253914

```

ddmcp.34='2103';ddmde.34='RDBINTTKN - RDB Interrupt Token'	00254014
ddmcp.35='11E8';ddmde.35='IPADDR - TCP/IP Address'	00254114
ddmcp.36='2113';	00254214
ddmde.36= ,	00254314
'PKGNAMECSN - RDB Package Name, Consistency Token, and Section Number'	00254414
ddmcp.37='11E8';ddmde.37='IPADDR - TCP/IP Address'	00254514
ddmcp.38='106D';ddmde.38='ACCSEC - Access Security'	00254614
ddmcp.39='2001';ddmde.39='ACCRDB - Access RDB'	00254714
ddmcp.40='210F';ddmde.40='RDBACCCL - RDB Access Manager Class'	00254814
ddmcp.41='2135';ddmde.41='CRRTKN - Correlation Token'	00254914
ddmcp.42='2104';ddmde.42='PRDDTA - product Specific Data'	00255014
ddmcp.43='213B';ddmde.43='TRGDFTRT - Target Default Value Return'	00255114
ddmcp.44='2414';ddmde.44='SQLSTT - SQL Statement'	00255214
ddmcp.45='0000';ddmde.45='DATA - CODE POINT = 0000'	00255314
ddmcp.46='2114';ddmde.46='QRYBLKSZ - Query Block Size'	00255414
ddmcp.47='2141';ddmde.47='MAXBLKEXT - Maximum Number of Extra Blocks'	00255514
ddmcp.48='214B';ddmde.48='DYNDTAFMT - Dynamic Data Format'	00255614
ddmcp.49='2137';ddmde.49='MEDDTASZ - Maximum Size of Medium Data'	00255714
ddmcp.50='2102';ddmde.50='QRYPRCTYP - Query Protocol Type'	00255814
ddmcp.51='2150';ddmde.51='QRYATTUPD - Query Attribute for Updatability'	00255914
ddmcp.52='215B';ddmde.52='QRYATTUPD - Query Attribute for Updatability'	00256014
ddmcp.53='215B';ddmde.53='QRYINSID - Query Instance Identifier'	00256114
ddmcp.54='215F';ddmde.54='QRYBLKFCT - Query Blocking Factor'	00256214
ddmcp.55='2115';ddmde.55='UOWDSP - Unit of Work Disposition'	00256314
ddmcp.56='220C';ddmde.56='ENDUOWRM - End Unit of Work Condition'	00256414
ddmcp.57='11DD';ddmde.57='TCPHOST - TCP/IP Domain Qualified Host Name'	00256514
ddmcp.58='106F';ddmde.58='SYNCLG - Sync Point Log'	00256614
ddmcp.59='211A';ddmde.59='RDBALWUPD - RDB Allow Updates'	00256714
ddmcp.60='1055';ddmde.60='SYNCCCTL - Sync Point Control Request'	00256814
ddmcp.61='2014';ddmde.61='EXCSQLSET - Set SQL Environment'	00256914
ddmcp.62='2116';ddmde.62='RTNSQLDA - Return SQL Descriptor Area'	00257014
ddmcp.63='119C';ddmde.63='CCSIDSBBC - CCSID for Single-byte Characters'	00257114
ddmcp.64='2001';ddmde.64='ACCRDB - Access RDB'	00257214
ddmcp.65='1055';ddmde.65='SYNCCCTL - Sync Point Control Request'	00257314
ddmcp.66='2014';ddmde.66='EXCSQLSET - Set SQL Environment'	00257414
ddmcp.67='200D';ddmde.67='PRPSQLSTT - PrepareSQL Statement'	00257514
ddmcp.68='2408';ddmde.68='SQLCARD - SQL Communications Area Reply Data'	00257614
ddmcp.69='200C';ddmde.69='OPNQRY - Open Query'	00257714
ddmcp.70='2006';ddmde.70='CNTQRY - Continue Query'	00257814
ddmcp.71='1248';ddmde.71='SYNCCRD - Sync Point Control Reply'	00257914
ddmcp.72='2121';ddmde.72='STTDECEL - Statement Decimal Delimiter'	00258014
ddmcp.73='1187';ddmde.73='SYNCTYPE - Sync Point Operation Type'	00258114
ddmcp.74='2146';ddmde.74='TYPSQLDA - Type of SQL Descriptor Area'	00258214
ddmcp.75='1801';ddmde.75='XID - Global Transaction Identifier'	00258314
ddmcp.76='1904';ddmde.76='XARETVAL - XA Return Value'	00258414
ddmcp.77='2148';ddmde.77='RTNEXTDTA - Return of EXTDTA Option'	00258514
ddmcp.78='1903';ddmde.78='XAFLAGS - XA Flags'	00258614
ddmcp.79='200B';ddmde.79='EXCSQLSTT - Execute SQL Statement'	00258714
ddmcp.80='2218';ddmde.80='RDBUPDRM - RDB Update Reply Message'	00258814
ddmcp.81='119F';ddmde.81='RLSCONV - Release Conversation'	00258914
ddmcp.82='2218';ddmde.82='RDBUPDRM - RDB Update Reply Message'	00259014
ddmcp.83='2450';ddmde.83='SQLATTR - SQL Statement Attributes'	00259114
ddmcp.84='2105';ddmde.84='RDBCMTOK - RDB Commit Allowed'	00259214
ddmcp.85='2005';ddmde.85='CLSQRY - Close Query'	00259314
ddmcp.86='2136';ddmde.86='SMLDTASZ - Maximum Size of Small Data'	00259414
ddmcp.87='2111';ddmde.87='OUTEXP - Output Expected'	00259514
ddmcp.88='1186';ddmde.88='FORGET - Forget Unit of Work'	00259614
ddmcp.89='2134';ddmde.89='QRYEXTDTASZ - Query Externalized Data Size'	00259714
ddmcp.90='215D';ddmde.90='QRYCLSIMP - Query Close Implicit'	00259815
ddmct = 90	00260016

```

/* read traces files into traces stem */
EXECIO * DISKR TRACES (STEM traces.)

say '--> DB2 Subsystem:' DB2
say '--> Trace type:' TraceType

/* Important: this example parses the GTF header, if using SMF data,
you need to copy the records to a FB dataset and map the SMF header*/

If TraceType = "GTF" then QWGTEID = x2c('EFB9')

say '--> IFCID:' IFCID "('X' || d2x(IFCID) || '')"
say '--> TRACES files contains' traces.0 'lines.'

Do i=1 to traces.0

call debug(copies('-',130))
call debug('Working with TRACE record' i)
call debug(copies('-',130))

if substr(traces.i,11,2) = QWGTEID then do

/*-----*/
/* GTF header section */
/*-----*/
/* get jobname */
QWGTJOB = substr(traces.i,17,8)
call debug("QWGTJOB" QWGTJOB)

/* get SSID */
QWGTSSID = substr(traces.i,29,4)
call debug("QWGTSSID" QWGTSSID)

/* get sequence number */
QWGTWSEQ = substr(traces.i,33,4)
QWGTWSEQc = c2d(QWGTWSEQ)
call debug("QWGTWSEQ" QWGTWSEQc)

/* get data length */
QWGTDLEN = substr(traces.i,25,2)
QWGTDLENC = c2d(QWGTDLEN)
call debug("QWGTDLEN" QWGTDLENC)

/* spanned record processing */
QWGTDSCC = substr(traces.i,27,1)
QWGTDSCCc = c2d(QWGTDSCC)
call debug("QWGTDSCC" QWGTDSCCc)

if QWGTDSCCc = '0' then do
call debug("Record not spanned")
recordcnt = recordcnt + 1
record.recordcnt = substr(traces.i,37,QWGTDLENC)
record.recordcnt = substr(traces.i,1,(QWGTDLENC+24))
call debug ("Not spanned >>" || record.recordcnt || "<<")
end
else do
if QWGTDSCCc = '1' then do
call debug("Working with first segment of record" QWGTWSEQc)

```

```

        recordcnt = recordcnt + 1                                00266114
/*                                                                    00266214
If it is the first segment (that is, QWGTDS01), save the entire 00266314
record including the sequence number (QWGTWSEQ) and the subsystem ID 00266414
(QWGTSSID).                                                       00266514
*/                                                                    00266614
        segment = substr(traces.i,25,QWGTDLenc)                  00266714
        segment = substr(traces.i,1,(QWGTDLenc+24))              00266814
        call debug ("first segment >>" || segment || "<<")      00266914
        record.recordcnt = segment                                00267014
/* process next segment of same sequence record */              00267114
/* assumption: next segments of the same sequence                00267214
   are written in increasing order, an initial sort of          00267314
   segments by QWGTWSEQ | QWGTDS01 may be needed */            00267414
                                                                    00267514
currQWGTWSEQ = QWGTWSEQc /* store current sequence */          00267614
do while ((QWGTWSEQc = currQWGTWSEQ) & (QWGTDS01 <> '2' ))    00267714
    i = i + 1                                                    00267814
                                                                    00267914
    /* get sequence number */                                    00268014
    QWGTWSEQ = substr(traces.i,33,4)                             00268114
    QWGTWSEQc = c2d(QWGTWSEQ)                                    00268214
    call debug("  QWGTWSEQ" QWGTWSEQc)                          00268314
                                                                    00268414
    /* get data length */                                       00268514
    QWGTDLenc = substr(traces.i,25,2)                           00268614
    QWGTDLenc = c2d(QWGTDLenc)                                  00268714
    call debug("  QWGTDLenc" QWGTDLenc)                         00268814
                                                                    00268914
    /* spanned record processing */                              00269014
    QWGTDS01 = substr(traces.i,27,1)                            00269114
    QWGTDS01c = c2d(QWGTDS01)                                   00269214
    call debug("  QWGTDS01" QWGTDS01c)                         00269314
                                                                    00269414
    segment = substr(traces.i,37,QWGTDLenc)                    00269514
    call debug ("next segment >>" || segment || "<<")          00269614
    record.recordcnt = record.recordcnt || ,                    00269714
        substr(traces.i,37,QWGTDLenc)                          00269814
                                                                    00269914
    end /*do while ((QWGTWSEQc = currQWGTWSEQ) & (QWGTDS01... */ 00270014
end /* if QWGTDS01 = x'00' then do */                            00270114
end                                                                00270214
end                                                                00270314
End /* Do i=1 to traces.0 */                                     00270414
                                                                    00270514
/*-----*/                                                    00270614
/* report records found after spanned processing */            00270714
/*-----*/                                                    00270814
say '--> Trace records available after proc. spanned:' recordcnt 00270914
say                                                                00271014
                                                                    00271114
/* write traces to file for debugging */                       00271214
do v = 1 to recordcnt                                           00271314
    queue record.v                                              00271414
end                                                            00271514
                                                                    00271614
/*                                                                00271717
"EXECIO" QUEUED() "DISKW ASSEMB (FINIS"                      00271817
*/                                                            00271914
                                                                    00272017

```

```

/*-----*/
/* process records */
/*-----*/

do x = 1 to recordcnt

    call debug(copies('-',50))
    call debug("Working with consolidated record" x)
    call debug(copies('-',50))

    call debug("--> length" length(record.x))
    call debug(">>>" || record.x || "<<<")

    /* get jobname */
    QWGTJOB = substr(record.x,17,8)

    /* is the record IFCID 180 ? */
    QWHSIID = ""
    /* offset to product section */
    off2PSect = c2d(substr(record.x,37,4))
    call debug("off2PSect" off2PSect)
    QWHSIID = substr(record.x,(off2PSect+1),2)
    QWHSIIDc = c2d(QWHSIID)
    call debug("QWHSIIDc (IFCID)" QWHSIIDc)

    /* warning --> this example only works with ifcid 180
       you may consider to generalize this program by
       mapping the product section */

    /* event type */
    QW0180E = ""
    QW0180E = substr(record.x,53,1)
    call debug("record" x "QW0180E" QW0180E)

    /* QW0180DL (H) SIZE OF QW0180DS VARIABLE LENGTH */
    QW0180DL = ""
    QW0180DL = substr(record.x,79,2)
    QW0180DLc = c2d(QW0180DL)
    toprocess = QW0180DLc
    toprocess = QW0180DLc - 14
    call debug("Record" x "QW0180DL" QW0180DLc)

    /* more --> refer to hlq.SDSNMACS(DSNDQW02) for more mapping */

    if ((QW0180E = "R") | (QW0180E = "S") & QWHSIIDc = "180") then do

        call debug('Record #' || x)

        if QW0180E = "R" then flow = "(REC) ==> "
        if QW0180E = "S" then flow = "<==== (SND) "
        if QW0180E = "R" then flow2 = "(REC) ----> "
        if QW0180E = "S" then flow2 = "<---- (SND) "

        if QW0180E = "R" then flow = QWGTJOB "(REC) ==> "
        if QW0180E = "S" then flow = QWGTJOB "<==== (SND) "
        if QW0180E = "R" then flow2 = QWGTJOB "(REC) ----> "
        if QW0180E = "S" then flow2 = QWGTJOB "<---- (SND) "

        call report("----> working with record" x "of type " QW0180E )
    end if
end do

```

00272114  
 00272214  
 00272314  
 00272414  
 00272514  
 00272614  
 00272714  
 00272814  
 00272914  
 00273014  
 00273114  
 00273214  
 00273314  
 00273414  
 00273514  
 00273614  
 00273714  
 00273814  
 00273914  
 00274014  
 00274114  
 00274214  
 00274314  
 00274414  
 00274514  
 00274614  
 00274714  
 00274814  
 00274914  
 00275014  
 00275114  
 00275214  
 00275314  
 00275414  
 00275514  
 00275614  
 00275714  
 00275814  
 00275914  
 00276014  
 00276114  
 00276214  
 00276314  
 00276414  
 00276514  
 00276614  
 00276714  
 00276814  
 00276914  
 00277014  
 00277114  
 00277214  
 00277314  
 00277414  
 00277514  
 00277614  
 00277714  
 00277814  
 00277914  
 00278314



```

offset = 81  /* to first len element */
/*-----*/
/* DRDA processing */
/*-----*/
do while (toprocess > 0)
  offset = offset + 6 /* offset to len of element */
  call debug("Offset at start Main loop" offset)
  len = substr(record.x,offset,2)
  lenc = c2d(len)
  call debug("==> lenc" lenc)
  cpointoff = offset + 2
  cpoint = substr(record.x,cpointoff,2)
  cpointc= c2x(cpoint)
  term = "UNKNOWN DRDA TERM - CODE POINT" cpointc "SKIP"
  do z = 1 to ddmct
    if ddmcp.z = cpointc then do
      term = ddmde.z
    end
  end
  call debug("==> cpoint" cpointc)

  /* print flow and term name */
  say flow term

  if substr(term,1,7) = 'UNKNOWN' then do
    call debug('skip record because of unknown term')
    leave
  end

  /* process inside data */
  /* ----- */
  woffs = cpointoff + 2
  wlenc = lenc - 4
  cnti = 0

  /* special cases */

  /* END special cases */

  do while (wlenc <> 0)
    cnti = cnti + 1
    call debug("woffs" woffs)
    currlen = substr(record.x,woffs,2)
    currlenc = c2d(currlen)
    call debug("currlenc" currlenc)
    woffs = woffs + 2
    call debug("woffs" woffs)
    currcpo = substr(record.x,woffs,2)
    currcpoc = c2x(currcpo)
    call debug("currcpoc" currcpoc)
    term2 = "UNKNOWN DRDA TERM - CODE POINT" currcpoc
    do z = 1 to ddmct
      if ddmcp.z = currcpoc then do
        term2 = ddmde.z
      end
    end
    say flow2 term2
    if substr(term2,1,7) = 'UNKNOWN' then do

```

```

00278414
00278514
00278614
00278714
00278814
00278914
00279014
00279114
00279214
00279314
00279414
00279514
00279614
00279714
00279814
00279914
00280014
00280114
00280214
00280314
00280414
00280514
00280614
00280714
00280814
00280914
00281014
00281114
00281214
00281314
00281414
00281514
00281614
00281714
00281814
00281914
00282014
00282114
00282214
00283414
00283514
00283614
00283714
00283814
00283914
00284014
00284114
00284214
00284314
00284414
00284514
00284614
00284714
00284814
00284914
00285014
00285114
00285214
00285314
00285414

```

```

        call debug('Skip process because unknown term')
        leave
    end
    /* special considerations for DATA:
        len is 0, need to use upper level len
        --> len and offset needs to be adapted */
    woffs = woffs + 2
    call debug("woffs" woffs) /* ok for DATA */

    if currcpoc <> '0000' then do

        /*following line added temporary */
        currlenc = currlenc - 2
        /*end following line added temporary */

        currrdata = substr(record.x,woffs,(currlenc-2))
    end
    else do
        /* adapt length for DATA */
        currlenc = lenc -4
        currrdata = substr(record.x,woffs,(currlenc-2))
    end

    call debug( "Current data" currrdata)
    /*-----*/
    /* print data elements */
    /*-----*/
    if DataOption = 'DATA=Y' & currcpoc <> '0000' ,
    then do
        margin = copies(' ',13)
        raw = ''
        ebc = ''
        asc = ''
        dcnt = 0
        rdcnt = 0
        title = '          +-- H E X -----' || ,
        '-----' || ,
        '+  +-- EBCDIC -----+  +-- ASCII -----+'
        bottom= '          +-----' || ,
        '-----' || ,
        '+  +-----+  +-----+'
        say title

        do pd = 1 to length(currrdata)
            dcnt = dcnt + 1
            rdcnt = rdcnt + 1
            d = substr(currrdata,pd,1)
            dc2x = c2x(d)
            dasc = translate(x2c(c2x(d)),toASCII,hextable)
            dasc = translate(x2c(c2x(d)),toASCII,hextable)
            if rdcnt = 9 then do
                rdcnt = 0
                raw = raw || ' ' || dc2x
                ebc = ebc || d
                asc = asc || dasc
            end
            else do
                if dcnt > 1 then do
                    raw = raw || ' ' || dc2x
                    ebc = ebc || d
                end
            end
        end
    end

```

```

        asc = asc || dasc
    end
    else do
        raw = dc2x
        ebc = ebc || d
        asc = asc || dasc
    end
end
if dcnt = 16 then do
    dcnt = 0
    rdcnt = 0
    line = margin || ' ' || raw || ' ' || ' ' ||,
           ' ' || ebc || ' ',
           || ' ' || ' ' || asc || ' '
    say line
    raw = ''
    ebc = ''
    asc = ''
end
end /* do pd = 1 to length(currdata) */
do w = (dcnt+1) to 16
    dummy2 = '..'
    dummy = '.'
    if w = 9 then do
        raw = raw || ' ' || dummy2
        ebc = ebc || dummy
        asc = asc || dummy
    end
    else do
        if w = 1 then do
            raw = raw || dummy2
            ebc = ebc || dummy
            asc = asc || dummy
        end
        else do
            raw = raw || ' ' || dummy2
            ebc = ebc || dummy
            asc = asc || dummy
        end
    end
end
end
line = margin || ' ' || raw || ' ' || ' ' ||,
       ' ' || ebc || ' ',
       || ' ' || ' ' || asc || ' '
say line
say botton
say
end
/*-----*/
/* END print data elements */
/*-----*/

woffs = woffs + (currlenc-4)
call debug("woffs" woffs)
wlenc = wlenc - currlenc
call debug("wlenc" wlenc)
call debug(" >>> end interaction number" cnti)
end
call debug("----->>>>> gone out interaction ")

```

```

call debug("----->>>>> record legth" length(record.x))      00297514
offset = woffs                                                    00297614
call debug("new offset" offset)                                    00297714
toprocess = toprocess - lenc                                       00297814
                                                                    00297914
call debug("==> curr cpointc  " cpointc  )                       00298014
call debug("==> new toprocess  " toprocess)                       00298114
                                                                    00298214
end /* process main record */                                      00298314
/*-----*/                                                       00298414
/* END DRDA processing */                                         00298514
/*-----*/                                                       00298614
                                                                    00298714
End /* if ((QW0180E = "R") | (QW0180E = "s")) then do */         00298814
Else do                                                            00298914
    call report("---> record" x "of type " QW0180E "skipped")     00299014
End                                                                00299114
                                                                    00299214
end /* do x = 1 to recordcnt */                                    00299314
                                                                    00299414
exit                                                              00299514
/*-----*/                                                       00299614
/* routines */                                                    00299714
/*-----*/                                                       00299814
debug:                                                            00299914
    arg message                                                    00300014
    if debug_flag = "y" then say ">>>" message                   00300114
return                                                            00300214
report:                                                            00300314
    arg message                                                    00300414
    if debug_flag = "y" then say "***  " message                 00300517
return                                                            00300614
                                                                    00300714
                                                                    00300814
codepoint2:                                                       00300914
/* define drda ddm code points and description */                00301014
arg ccc                                                            00301114
do z = 1 to ddmct                                                  00301214
    if ddmcp.z = ccc then do                                       00301314
        tercc = ddmde.z                                          00301414
    end                                                            00301514
end                                                                00301614
say flow2 tercc                                                    00302014
return                                                            00310010

```

---

# Abbreviations and acronyms

<b>ACR</b>	Automatic client reroute	<b>DB2RA</b>	DB2 Request Access
<b>AES</b>	Advanced Encryption Standard	<b>DBAT</b>	Database access thread
<b>AIX</b>	Advanced Interactive eXecutive from IBM	<b>DBD</b>	Database descriptor
<b>APAR</b>	Authorized program analysis report	<b>DBID</b>	Database identifier
<b>AR</b>	Application Requester	<b>DBMS</b>	Database management system
<b>ARM</b>	Automatic restart manager	<b>DBRM</b>	Database request module
<b>AS</b>	Application Server	<b>DCL</b>	Data control language
<b>ASCII</b>	American National Standard Code for Information Interchange™	<b>DCS</b>	Database Connection Services
<b>BLOB</b>	Binary large objects	<b>DDCS</b>	Distributed database connection services
<b>BSDS</b>	Boot strap data set	<b>DDF</b>	Distributed Data Facility
<b>CA</b>	Certification Authority	<b>DDL</b>	Data Definition Language
<b>CA</b>	Coordinator Agent	<b>DDM</b>	Distributed Data Management
<b>CCA</b>	Client configuration assistant	<b>DLL</b>	Dynamic load library manipulation language
<b>CCSID</b>	Coded character set identifier	<b>DML</b>	Data manipulation language
<b>CD</b>	Compact disk	<b>DMZ</b>	Demilitarized zone
<b>CDB</b>	Communication database	<b>DNS</b>	Domain name server
<b>CDRA</b>	Character Data Representation Architecture	<b>DRDA</b>	Distributed Relational Database Architecture
<b>CEC</b>	Central electronics complex	<b>DSC</b>	Dynamic statement cache, local or global
<b>CF</b>	Coupling facility	<b>DTT</b>	Declared temporary tables
<b>CFCC</b>	Coupling facility control code	<b>DUW</b>	Distributed Unit of Work
<b>CFRM</b>	Coupling facility resource management	<b>DVIPA</b>	Dynamic VIPA
<b>CGI</b>	Common Gateway Interface	<b>EA</b>	Extended addressability
<b>CICS</b>	Customer Information control System	<b>EAR</b>	Enterprise Archive
<b>CLI</b>	Call level interface	<b>EBCDIC</b>	Extended binary coded decimal interchange code
<b>CLP</b>	Command line processor	<b>ECS</b>	Enhanced catalog sharing
<b>CPU</b>	Central Processing Unit	<b>ECSA</b>	Extended common storage area
<b>CRM</b>	Customer relationship management	<b>EDM</b>	Environment descriptor management
<b>CS</b>	Cursor stability	<b>ERM</b>	Enterprise resource management
<b>CSA</b>	Common storage area	<b>ERP</b>	Enterprise resource planning
<b>CST</b>	Consolidated Service Test	<b>ESA</b>	Enterprise Systems Architecture
<b>CSV</b>	Comma separated value	<b>ESP™</b>	Enterprise Solution Package
<b>CTT</b>	Created temporary table	<b>ETR</b>	External throughput rate, an elapsed time measure, focuses on system capacity
<b>DASD</b>	Direct access storage device	<b>FD:OCA</b>	Formatted Data Object Content
<b>DB2 Connect EE</b>	DB2 Connect Enterprise Edition		
<b>DB2 Connect PE</b>	DB2 Connect Personal Edition		
<b>DB2 PE</b>	DB2 Performance Expert		

<b>FIPS</b>	Federal Information Processing Standards	<b>JNDI</b>	Java Naming and Directory Interface
<b>FTD</b>	Functional track directory	<b>JPA</b>	Java Persistence API
<b>FTP</b>	File Transfer Program	<b>JTA</b>	Java Transaction API
<b>GB</b>	Gigabyte (1,073,741,824 bytes)	<b>JTS</b>	Java Transaction Service
<b>GBP</b>	Group buffer pool	<b>JVM</b>	Java Virtual Machine
<b>GRS</b>	Global resource serialization	<b>KAS</b>	Kerberos authentication server
<b>GSKit</b>	Global security kit	<b>KDB</b>	Kerberos database
<b>GUI</b>	Graphical user interface	<b>KDC</b>	Key distribution center
<b>GTF</b>	Generalized Trace Facility	<b>LA</b>	Logical Agent
<b>HFS</b>	Hierarchical file system	<b>LDAP</b>	Lightweight Directory Access Protocol
<b>HPJ</b>	High performance Java	<b>LE</b>	Language Environment
<b>HTTP</b>	Hypertext Transfer Protocol	<b>LOB</b>	Large object
<b>I/O</b>	Input/output	<b>LPAR</b>	Logical partition
<b>IBM</b>	International Business Machines Corporation	<b>LPL</b>	Logical page list
<b>ICF</b>	Integrated coupling facility	<b>LRECL</b>	Logical record length
<b>ICF</b>	Integrated catalog facility	<b>LRSN</b>	Log record sequence number
<b>ICMF</b>	Internal coupling migration facility	<b>LUW</b>	Logical unit of work
<b>ICMP</b>	Internet Control Message Protocol	<b>LVM</b>	Logical volume manager
<b>ICSF</b>	Integrated Cryptographic Service Facility	<b>MB</b>	Megabyte (1,048,576 bytes)
<b>IDS</b>	Informix Dynamic Server	<b>MTS</b>	Microsoft Transaction Server
<b>IETF</b>	Internet Engineering Task Force	<b>NPI</b>	Non-partitioning index
<b>IFC</b>	Instrumentation facility component	<b>ODB</b>	Object descriptor in DBD
<b>IFCID</b>	Instrumentation facility component identifier	<b>ODBC</b>	Open Data Base Connectivity
<b>IFI</b>	Instrumentation Facility Interface	<b>OMEGAMON PE</b>	IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS
<b>IPCS</b>	interactive problem control system	<b>PAV</b>	Parallel access volume
<b>IPLA</b>	IBM Program Licence Agreement	<b>PDS</b>	Partitioned data set
<b>IPsec</b>	IP security	<b>PIB</b>	Parallel index build
<b>IRLM</b>	Internal resource lock manager	<b>PSID</b>	Pageset identifier
<b>ISPF</b>	Interactive system productivity facility	<b>PSP</b>	Preventive service planning
<b>ISV</b>	Independent software vendor	<b>PTF</b>	Program temporary fix
<b>IT</b>	Information Technology	<b>PUNC</b>	Possibly uncommitted
<b>ITR</b>	Internal throughput rate, a processor time measure, focuses on processor capacity	<b>QA</b>	Quality Assurance
<b>ITSO</b>	International Technical Support Organization	<b>QMF</b>	Query Management Facility
<b>IVP</b>	Installation verification process	<b>RACF</b>	Resource Access Control Facility
<b>JCE</b>	Java Cryptography Extension	<b>RBA</b>	Relative byte address
<b>JDBC</b>	Java Database Connectivity	<b>RDBMS</b>	Relational Database Management Systems
<b>JFS</b>	Journalled file systems	<b>RECFM</b>	Record format
		<b>RID</b>	Record identifier
		<b>ROT</b>	Rule of thumb
		<b>RR</b>	Repeatable read
		<b>RRS</b>	Resource recovery services

<b>RRSAF</b>	Resource recovery services attach facility
<b>RS</b>	Read stability
<b>RUW</b>	Remote Unit of Work
<b>SAA</b>	System Application Architecture
<b>SAFR</b>	Scalable Architecture for Financial Reporting
<b>SAI</b>	Secondary authorization ID
<b>SCA</b>	Shared communication area
<b>SD</b>	Sysplex distributor
<b>SDK</b>	Software developers kit
<b>SDM</b>	System data mover
<b>SDSF</b>	System Display and Search Facility
<b>SMF</b>	System Management Facility
<b>SMIT</b>	System Management Interface Tool
<b>SNA</b>	Systems Network Architecture
<b>SRB</b>	Service Request Block
<b>SRM</b>	System Resource Manager
<b>SU</b>	Service Unit
<b>SSL</b>	Secure Sockets Layer
<b>TCO</b>	Total cost of ownership
<b>TGS</b>	Ticket granting server
<b>TNG</b>	Transaction Name Group
<b>UDP</b>	User Datagram Protocol
<b>UOW</b>	Unit Of Work
<b>USS</b>	UNIX System Service
<b>VIPA</b>	Virtual IP Addressing
<b>VPN</b>	Virtual Private Network
<b>VTAM</b>	Virtual Telecommunication Access Method
<b>WAS</b>	WebSphere Application Server
<b>WLB</b>	workload balancing
<b>WLM</b>	Workload Manager
<b>XA</b>	Extended Architecture
<b>zAAP</b>	z Application Assist Processor
<b>zIIP</b>	z Integrated Information Processor

Archived



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 463. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *LOBs with DB2 for z/OS: Stronger and Faster*, SG24-7270
- ▶ *Securing DB2 and Implementing MLS on z/OS*, SG24-6480
- ▶ *HiperSockets Implementation Guide*, SG24-6816
- ▶ *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7535
- ▶ *IBM z/OS V1R10 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7699
- ▶ *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083
- ▶ *A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS*, SG24-7224
- ▶ *Data Federation with IBM DB2 Information Integrator V8.1*, SG24-7052
- ▶ *Publishing IMS and DB2 Data using WebSphere Information Integrator: Configuration and Monitoring Guide*, SG24-7132
- ▶ *DB2 9 for z/OS: Packages Revisited*, SG24-7688
- ▶ *DB2 for z/OS: Considerations on Small and Large Packages*, REDP-4424
- ▶ *Effective zSeries Performance Monitoring Using Resource Measurement Facility*, SG24-6645
- ▶ *DB2 for z/OS and OS/390 : Squeezing the Most Out of Dynamic SQL*, SG24-6418
- ▶ *System Programmer's Guide to: Workload Manager*, SG24-6472-00

## Other publications

These publications are also relevant as further information sources:

- ▶ *DB2 Version 9.1 for z/OS Administration Guide*, SC18-9840-04
- ▶ *DB2 Version 9.1 for z/OS Data Sharing: Planning and Administration*, SC18-9845-02
- ▶ *DB2 Version 9.1 for z/OS Reference for Remote DRDA Requesters and Servers*, SC18-9853-02
- ▶ *DB2 Version 9.1 for z/OS Installation Guide*, GC18-9846-07
- ▶ *DB2 Version 9.1 for z/OS Command Reference*, SC18-9844-03
- ▶ *DB2 Version 9.1 for z/OS Application Programming and SQL Guide*, SC18-9841-04
- ▶ *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851-05

- ▶ *DB2 Version 9.1 for z/OS SQL Reference*, SC18-9854-05
- ▶ *DB2 Version 9.1 for z/OS ODBC Guide and Reference*, SC18-9850-02
- ▶ *DB2 Version 9.1 for z/OS Application Programming Guide and Reference for Java*, SC18-9842-04
- ▶ *DB2 Version 9.1 for z/OS Utility Guide and Reference*, SC18-9855-03
- ▶ *DB2 Version 9.1 for z/OS Codes*, GC18-9843-05
- ▶ *DB2 Version 9.1 for z/OS Messages*, GC18-9849-05
- ▶ *DB2 9.1 Administration Guide: Implementation*, SC10-4221
- ▶ *DB2 Connect Version 9 Quick Beginnings for DB2 Connect Servers*, GC10-4243-00
- ▶ *DB2 Connect Version 9 User's Guide*, SC10-4229-00
- ▶ *z/OS V1R10.0 MVS Initialization and Tuning Reference*, SA22-7592
- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, Report Reference*, SC18-9984
- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, Report Command Reference*, SC18-9985
- ▶ *z/OS V1R10.0 Security Server RACF System Programmer's Guide*, SA22-7681-10
- ▶ *z/OS V1R10.0 SDSF Operation and Customization*, SA22-7670-11
- ▶ *z/OS V1R10.0 MVS Planning: Workload Management*, SA22-7602-00
- ▶ *z/OS V1R10.0 Network Authentication Service Administration*, SC24-5926-07
- ▶ *z/OS V1R10.0 Communication Server IP: Diagnosis Guide*, GC31-8782-09
- ▶ *z/OS V1R10 Communications Server IP: Configuration Guide*, SC31-8775-14
- ▶ *z/OS V1R10 Communications Server IP: Configuration Reference*, SC31-8776-15
- ▶ *z/OS V1R10 Communications Server IP: System Administrator's Commands*, SC31-8781-08
- ▶ *z/OS V1R10 Security Server RACF Command Language Reference*, SA22-7687-12

## Online resources

These Web sites are also relevant as further information sources:

- ▶ The Open Group Web site  
<http://www.opengroup.org/dbiop/>
- ▶ IBM System z9 Enterprise Class  
<http://www.ibm.com/systems/z/hardware/z9ec/specifications.html>
- ▶ the IBM DB2 for Linux, UNIX, and Windows Information Center  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>
- ▶ IBM DB2 Connect V9.7 for Linux, UNIX, and Windows, release  
<http://news.prnewswire.com/DisplayReleaseContent.aspx?ACCT=104&STORY=/www/story/04-22-2009/0005011001&EDATE=>
- ▶ IBM Data Server Clients and Drivers overview  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.client.doc/doc/c0023452.html>

- ▶ *DRDA V4, Vol. 1: Distributed Relational Database Architecture*, available from:  
<http://www.opengroup.org/onlinepubs/9699939399/toc.pdf>
- ▶ *DRDA V4, Vol. 2: Formatted Data Object Content Architecture*, available from:  
<http://www.opengroup.org/onlinepubs/9699939299/toc.pdf>
- ▶ *DRDA V4, Vol. 3: Distributed Data Management Architecture*, available from:  
<http://www.opengroup.org/onlinepubs/9699939199/toc.pdf>
- ▶ *End-to-end federated trusted contexts in WebSphere Federation Server V9.5* by Priya Baliga and Eileen Lin, available from:  
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0712baliga/index.html>

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Redguides, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

Archived

# Index

## Numerics

2-Phase commit 8  
-919 40

## A

ACB 21  
Accounting trace 363  
ACCRDB 336–337  
ACCRDBRM 336–337  
ACR (Automatic client reroute) 230  
ADBAT 236, 272  
address space 35, 70, 131, 139, 234, 243  
    IRLM stand 18  
ADO 44, 145  
Advanced Encryption Standard 147  
AES (Advanced Encryption Standard) 130, 134, 147  
    encryption 133, 138, 147  
After 134  
agent pool 24, 258  
ageServerLists 238  
AIX (Advanced Interactive eXecutive) 27, 63, 122, 259, 305, 331  
ALIAS 37, 70, 80, 196, 236, 248  
Alias 82, 187, 238, 298  
analyzing the packet trace 352  
APAR II4293 396  
APAR PK80474 249  
APPC 107  
APPL 18, 235, 282  
application programming interface (API) 15  
application server  
    generic user ID 140  
    your copy 252  
Application Support Protocol 11  
Application-directed access 186  
AR (application requester) 11, 14, 70  
AS (application server) 11–12, 14, 24, 70, 140, 171  
ASCII 125, 159, 164, 334  
ASCII mode 161  
AT-TLS (Application Transparent Transport Layer service) 151  
AUTHENTICATION 124, 296  
Authentication 255  
authentication 137, 143, 299  
authentication mechanism 130–131, 133  
    insecure form 134  
authentication standard (AS) 130  
authorization 89, 95  
authorization ID 132

## B

batch job 197  
BIND PACKAGE options 191

Binding applications 190  
block cache 263  
BM Data Server Client 337  
BSDS (bootstrap data set) 35, 74, 76, 97, 153, 159, 245, 248, 318

## C

CA (certification authority) 152  
cache 131, 138, 148, 300  
captured SQL statement  
    next step 177  
catalog database  
    command 131  
    DB9C 257  
catalog tables 124, 195  
CCSID 14, 125, 337  
CDB (communication database) 20, 35, 73, 76, 86, 135, 139–140  
    table 139–140, 172  
CDRA (Character Data Representation Architecture) 12  
CERTAUTH Label 158  
character set 14, 125  
CICS 57  
CICS (Customer Information control System) xxvi, 107  
CLI (call level interface) 4, 34, 130, 142, 239, 253, 286, 296, 325  
    application 181  
    driver 131, 253  
    trace 325  
CLI (call level interface) trace 325  
CLIENT 130, 133, 296, 343  
client information 112, 142  
ClientAcctStr 346  
ClientApplName 346  
ClientUserid 346  
ClientWrkstnName 346  
CLP (Command Line Processor) 116, 200  
CLSQRY 337  
CMTSTAT 23, 90–91, 94, 106, 271, 274, 389  
CNTQRY 336  
COBOL xxvi, 139  
code points 337  
com.ibm.db2.jcc.DB2C onnection 144  
comma separated value 354  
command line  
    interface 44  
    processor 44, 169  
COMMIT 5–6, 94, 261, 273, 277, 373  
Commit phase 1 5  
Commit phase 2 6  
COMMON 331  
CONDBAT 90–91, 236, 271–272, 307  
configuration file 142–143, 145, 154, 252  
    DS driver 254

- CONNECT 86, 91, 158, 190, 260, 262, 337
  - Type(1) 189
  - Type(2) 189
- CONNECT TO 8, 187–188, 261–262
- connection concentration 48, 53
- connection concentrator 26, 53, 236, 242
- connection management 186
- connection pool 24
- connection request 62, 90, 92, 95, 130, 133–134, 241, 262
- connectivity 10, 52, 147, 241, 249, 314
- consolidation 60
- Control Center 45, 348
- cost 92, 284, 370
- CPU 17, 23, 56, 109–110, 181, 235, 274, 278, 345
- CREATE 196
- CREATE ALIAS 187
- CS (communication server) 151
- CSV 354, 360
- CTHREAD 91–92, 307
- CURRENTDATA 191
- CURRENTPACKAGESET 174, 180
- CURRENTSERVER 192, 309

## D

- D9C1 DSNLTDDF 84, 101–102, 236
- D9C1 failure 262
- d9cg.itso.ibm.com d9cg 81
- data access 69, 129, 176
- data distribution 8
- data sharing 36, 70, 75, 233–234, 239
- Data sharing group 312
- data source 24, 26, 42, 250, 338
- Database Access
  - Thread 92
- Database Access Thread 19
- database connection
  - Information 169
  - services 300
- Database Support Protocol 11
- DataPower 172
- DATE 157–158, 235, 366
- DB2
  - DATE 372
  - V8 APARs 395
- DB2 9 4, 41, 71, 85, 130, 138–139, 235, 259
  - catalog 195
  - DSNZPARM macro DSN6FAC 97
- DB2 accounting trace 284, 287, 343
- DB2 Administration Client 41
- DB2 Application Development Client 41
- DB2 client 41, 55, 135, 161, 178, 180, 259, 324
- DB2 Command Line Processor 174, 200, 325
- DB2 Connect 14, 33, 69, 116, 130, 238, 270, 324
  - Agent pooling 24
- DB2 Connect Client 52
- DB2 Connect Personal Edition 133
- DB2 Connect Server 239
- DB2 data 25, 37, 76–77, 95, 145, 234
  - distributed DVIPA 61

- IP address 55, 234
- member sysplex 248
- DB2 database
  - server 15
  - service 19
- DB2 Enterprise Server Edition 303
- DB2 family
  - compatibility 134
- DB2 for Linux, UNIX and Windows 14
- DB2 for System i 14
- DB2 for VSE and VM 15
- DB2 for z/OS 14, 69, 130, 234
- DB2 for z/OS server 97, 104, 138, 150–151, 234, 247
- DB2 Information Integrator 27
- DB2 installation parameters 90
- DB2 member 25, 55, 79
- DB2 Performance Expert 142, 286, 364
- DB2 Private Protocol 33, 105
- DB2 resource 55, 132
- DB2 Run-Time Client 41
- DB2 subsystem 17, 35, 71–72, 131, 136, 196, 240
  - additional address space 17
  - requests data 35
- DB2 support
  - DES encryption/decryption 147
- DB2 system 25, 69, 92
  - address space 108
- DB2 UDB
  - Version 7.1 27
- DB2 UDB Universal Driver for SQLJ and JDBC 15
- DB2Binder utility 104
- DB2CLI.INI 174–175, 206, 326
  - CURRENTPACKAGESET 175
- DB2CONNECT\_IN\_APP\_PROCESS 258, 295
- Db2drdat 333
- db2ds.setSecurityMechanism 148
- db2dsdriver 326
- db2dsdriver.cfg 206, 326
- db2level 303
- db2set 166, 258, 295, 332
- DB2START time 258
- DB9A DSNLTDDF 84, 169
- DB9C IPNAME 99, 101, 103
- DB9C9.SDSN Exit 79, 100
- DBAT 19, 22, 56, 92, 94, 263, 373
- DBMS 6, 125
- DBPROTCL 307
- DBPROTOCOL 309–310
- DBRM 285, 287
- DBRM name 195
- DCS (DATABASE command completed successfully) 169
- DCS (Database Connection Services) 168–169, 244, 300, 337
  - directory 257–258
- DD DSN 74, 154, 159
- DD SYSOUT 74, 155
- DDF (Distributed Data Facility) 17, 35, 70–71, 90, 131, 134, 159, 235, 271, 344
  - address space 35, 70, 85, 105, 156

- access 19
- output 70
- remote subsystem 19
- setup 103
- started procedure names 75
- startup 71
- implementation 18
- location 159
- network protocols 20
- report 84, 169
- request 97
- DDM (Distributed Data Management) 11–12, 463
  - command 14
- default qualifier 146
- DEFERPREP 310
- DEFERPREPARE 310
- defined dynamic virtual IP address 99
- DELETE 140, 307
- deployment scenario 249
- DES (Data Encryption Standards) 147
- Diffie-Hellman algorithm 147
- directory cache 138, 148–149
- DISABLE 90–91
- DISCONNECT 192, 199
- Discover 296
- DISPLAY 83–84, 169, 236, 272, 281, 343
- DISPLAY DDF 84, 236
- DISPLAY THREAD 319, 348
- DISPLAY VIRTSTOR 85–86
- DIST 18, 35, 85, 106, 131, 235
- distributed access threads 22
- distributed data topology 4
  - Distributed request 9
  - Distributed unit of work 8
  - Remote request 6
  - Remote unit of work 7
- distributed request 9
- distributed unit of work 8
- DISTSERV 112, 193, 278, 289, 319, 348
- DNS 72, 75, 102, 304
- domain name 35, 72
  - distributed directory 72
- domain name server (DNS) 72
- dormant 186
- DRDA (Distributed Relational Database Architecture) 3, 11, 33, 70–71, 129, 234, 270, 281, 463
  - access 126, 130, 141, 147, 234, 248
  - client 95–96, 130, 234
    - LUs 129
  - connection 23, 234, 243
  - implementations 14
  - Level 2 8
  - levels 24
  - port 73, 77, 245
  - trace 325
  - workload balancing 242
  - workloadbalancing 234
- DRDA AR
  - client 133, 241
  - client system 243
  - function 37, 71, 77, 135
  - platform 51, 130, 249
  - site 133
- DRDA AR client 133, 240, 242
- DRDA architecture
  - Functions and protocols 11
- DRDA Argon 130
- DRDA connection 131
- DRDA PORT 95
- Driv 227
- Drivers 227
- DS (database server) 10–12, 55, 73, 86, 135, 175, 239, 299–300, 323
- ds driver 148, 179, 240, 244
  - configuration file 253, 255
- DSN3@ATH 132
- DSN3@SGN 132
- DSN6SYSP 277
- DSNJU003 95, 100
- DSNJU004 98–99
- DSNL081I Status 84, 169
- DSNL082I Location 84, 169
- DSNL084I TCPPORT 84, 169, 236
- DSNL085I IPADDR 84, 169, 236
- DSNL086I RESYNC Domain 169
- DSNL089I MEMBER IPADDR 84–85, 236
- DSNL090I DT 236
- DSNL093I DSCDBAT 236
- DSNL099I DSNLTDDF 84, 169, 236
- DSNLEUSR 89, 139
- DSNR 131
- DSNRLST 310
- DSNTEP2 196
- DSNTIP5 90, 132, 307
- DSNTIPE 90–91, 271
- DSNTIPR 90, 134, 271, 307
- DSNZPARM 23, 35, 90, 132, 134, 271–272, 376
  - CONDBAT 273
  - DDF 90, 93
  - MAXDBAT 272, 274
  - PRGSTRIN 222
  - TCPALVER 132
- DSNZPARM parameter 93
  - name 93
- DUW 8
- DVIPA (Dynamic Virtual IP Addressing) 25, 55, 70, 75–76, 99, 234, 243
- dynamic SQL 15, 94, 129, 173, 175–176, 181, 332
  - application 173–174
  - main security concerns 173
  - OK 173
  - package 174
  - packages force 173
  - program 174–175
  - security concern 173
  - security implications 173
  - security issues 129
  - statement 178
  - static execution 176
  - vehicle 176

DYNAMICRULES 173

## E

EBCDIC 125, 159, 161, 334  
elated 265  
ENABLE 90–91, 296  
Enclave 235, 288, 344  
enclaves 105, 288  
Encryption 141  
ENDQRYRM 337  
ENDUOWRM 337  
environment 33, 69–70, 135–136, 233, 270, 395  
ESTBLSH 80, 83, 169  
Example 159, 271–272  
Excel 374  
EXCSAT 336  
EXCSATRD 336  
Export 354, 360  
Extended security 95, 134  
extract 127–128  
EXTRAREQ 90, 96, 307  
EXTRASRV 90, 96, 307  
EXTSEC 90, 95, 132, 134, 307

## F

FD  
OCA 12  
federated database 26–27  
Distributed request 26  
Federation 28–29  
following SQLCODEs  
message texts 146  
FOR 102, 104, 134, 236, 259, 272–274, 330  
FOR FETCH ONLY 179, 333  
Formatted Data Object Content Architecture 11–12, 463  
full-function product 45

## G

GB bar 20, 85  
GRANT 174, 320

## H

hierarchical file system (HFS) 75  
HyperSocket 58, 127, 280, 391  
HOME 70, 82, 154  
HOPAATH 90, 96  
HTTP 270

## I

I/O 20, 127, 235–236, 272, 289, 345  
IBM Data Server Client 15, 34, 41, 43, 45, 53  
IBM Data Server Driver for ODBC, CLI and .NET, Open-Source 15  
IBM DB2  
DataPropagator 31  
driver 41, 147, 249  
JDBC universal driver 249

IBM Informix Dynamic Server 14  
IBM Key Management  
tool 161  
tool window 162  
IBM Tivoli OMEGAMON XE for DB2 Performance Expert  
on z/OS 195  
ICSF 149  
ICSF (Integrated Cryptographic Service Facility) 139, 149  
IDTHTOIN 90, 94, 96, 271, 274, 307  
IEAOPTxx 282  
IEASYSxx 85  
IFCID 195  
1 380  
180 333  
3 288, 377  
II14203 70  
II4293 396  
IIPHONORPRIORITY 283  
IMS 57, 107, 307, 364  
INACONN 236  
inactive connection 22, 94  
inactive connection support 22  
inactive DBAT 94  
inactive DBATs  
high water mark 94  
inactive thread 94  
INADBAT 236  
INDOUBT 6, 373  
INFLIGHT 6  
in-flight unit 247, 261  
information 27, 41, 112, 135, 145, 279–280, 332  
information integration 28  
Informix 27, 31, 63, 340  
Informix Dynamic Server 31  
INSERT 139, 261, 264  
integration 27, 30, 234  
Internet Engineering Task Force 150  
IP address 20, 35, 71–72, 156, 159, 234, 236, 317  
IP security 141, 150  
IPCS 352, 355  
IPNAME 84, 103, 169, 236  
IPSec 150  
iSeries 31, 339  
ISOLATION 124, 310  
ISP 154  
IY99846 222, 397

## J

Java xxvi, 54, 112, 148, 249  
Java application 15, 144, 167–168, 249  
client information 144  
data access performance 16  
data stream encryption 150  
example 144  
security benefits 176  
server scenario configuration 250, 254  
server world 176  
SSL Connection 167  
static SQL 15



- step 178
- JCC (JAVA Common Client) 16
- JCL 71, 74, 159
- JDBC (Java Database Connectivity) 4, 34, 85, 104, 129, 142, 249, 325
  - application 175–177
  - driver
    - architecture 42
  - trace 338
- join 9, 280

## K

- KDC (Key Distribution Centre) 137
- KEEPDYNAMIC 23, 94, 253, 310
- Kerberos 96, 130–131, 133, 135, 137
  - authentication
    - mechanism 137
    - server 137
    - service 137
  - authentication server 137
  - database 137
- keystore 160
- keytool 161
- keytool command 159–160
  - shows sample output 161

## L

- Language Environment 70–71, 156
- LI682COLLID.LI682C 197
- LI682COLLID.LI682D 197
- Linux xxvi, 33, 128
- Linux, UNIX, and Windows (LUW) 4
- local DB2
  - data 33
  - database 34
  - subsystem 19, 104
- local system 287
- LOCATION Alias 93
- location alias
  - DB9CALIAS 248
  - DB9CSUBSET 248
  - name 248
- LOCATION name 35, 93, 97
- location name 6, 37, 86, 93, 195, 249
- location transparency 187
- LPAR 58, 80
- LU name 21, 93, 97
- LU6.2 21
- LUNAME 84, 103, 169, 236, 272, 318, 370
- LUW ESE 33
  - application requester 37
  - ARs 36
  - DB2 36
- LUW Version
  - 8 147
  - 9.5 FixPack 3 40
- LUWID 134, 274, 305, 315

## M

- maintenance 70, 395
- MAKESITE 75
- MATCH 180
- max number 251, 254
- MAXDBAT 90–91, 235, 271, 376
- Maximum Number 26
- maximum number 91
- MAXTYPE1 90, 94, 307
- MDBAT 236, 272
- member D9C1 82, 84, 241
- message 318, 337
- migration
  - PP to DRDA 194, 309
- monitor enclaves 288
- monitor trace 363
- multiple DBMSs 5
  - SQL statements 6
  - transaction concept 8

## N

- NCOMMIT 6
- network analyzer 361
- node directory 168, 170
- NULL RPORT 99, 101, 103
- NUMBER 140, 305, 365

## O

- OA27145 147, 397
- ODBC (Open Data Base Connectivity) 4, 34, 85, 129, 142, 325
  - application 15, 178
  - driver 143, 167, 325
- OLD D SN 98
- One DBMS 7
- open source 34
- OPNQRY 336
- OPNQRYRM 336
- OPTIONS 192, 197, 366
- Oracle 28

## P

- package owner 173–174
  - authorization limits 174
- Parallel Sysplex 36
  - environment 36, 75
  - technology 36
- parameter name 93, 143
- Part 2 128
- pass through 356
- PassTickets 96, 134
- PATH 146
- performance 53, 105, 172–173, 395
- performance analysis
  - Step-by-step 390
- performance trace 363
- Personal Edition 133, 259
- Phase 1 6

## Phase 2 6

PK03045 396  
PK03045) 235  
PK05198 265, 396  
PK06697 265, 396  
PK18454 285, 396  
PK25395 286, 396  
PK30450 396  
PK38867 235, 281, 396  
PK41236 253, 281, 396  
PK4166 224  
PK41661 97, 396  
PK44617 146, 396  
PK46079 91, 97, 222, 397  
PK47649 397  
PK50575 397  
PK51045 286, 397  
PK56287 147, 397  
PK56356 397  
PK59385 91, 97, 397  
PK62161 377, 397  
PK64045 397  
PK64298 397  
PK65367 398  
PK67794 312, 398  
PK68746 220, 398  
PK69659 64, 227, 398  
PK74330 347, 398  
PK75338 398  
PK76143 133, 398  
PK78553 199, 398  
PK79228 398  
PK79327 398  
PK79700 398  
PK80474 83, 236, 249, 317, 399  
PK81175 399  
PK85150 399  
PK89128 377, 397, 399  
PK89820 396, 399  
PKLIST 124, 198  
plan 95, 124  
policy agent 151–152  
    RACF resource 152–154  
    started task 154  
POOLINAC 91, 96, 271, 307  
port number 20, 72  
PP (private protocol) 33, 94, 105, 309, 386  
prddta 343  
Precompiler options 190  
presumed abort protocol 5  
presumed commit. 6  
presumed nothing 6  
PRGSTRIN 91, 97  
primary authorization ID (PAI) 180  
Program preparation 190  
PROJECTCPU 282  
props.setP roperty 144  
protocol 5, 35, 87, 94, 152, 299–300, 355  
PRPSQLSTT 336  
pureQuery 16, 176

## Q

QMF 91  
QRYDTA 10, 336  
QUEDBAT 236, 272

## R

RACDCERT 152  
RACDCERT CERTAUTH 157, 159  
RACDCERT ID 158  
RACF 70–71, 131  
    PassTickets 135  
RACF command 136, 157  
RACF ID 131  
RACF PassTicket 88, 96, 135–136  
RACF PassTickets 134–135  
RACF password 133, 135  
RACF resource 136, 153  
RDBCMM 337  
read 7, 71, 91, 313–314, 334  
RECTRACE 380  
Redbooks Web site 463  
    Contact us xxviii  
relational database  
    architecture 3  
    manager 13  
RELEASE 188  
release pending 186  
remote connection 19, 94  
remote DB2  
    location 73  
    subsystem 20, 136  
remote location 19, 73, 86  
remote request 6  
remote unit of work 7–8  
replication 31  
Resource Limit Facility 310  
response time goal 106  
RESYNC 77, 90, 94  
resync port 74, 79–80  
resynchronization port 35, 73, 75  
RETURN 91, 93, 95, 170  
REXX 195  
RLF 310–311  
RLFERRD 90, 94  
RMF 235, 288  
RMF Post Processor 288  
rollback 6, 94, 262  
RUW 7

## S

same host 72, 258  
SCPD9C1 Password 103  
seamless failover 246  
SECCHK 336, 360  
SECCHKRM 336  
secondary authorization ID (SAI) 132  
SECPORT 73, 84, 159, 168–169, 236  
security 10, 13, 71, 134, 137, 300, 336  
    Kerberos 137

- security mechanism (SECMEC) 134
- security option 87, 95, 129, 134
- SELECT 174, 179, 263
- SERVER 124, 130, 134, 236, 274, 289, 296, 348
- server 11, 34, 71–72, 129–130, 133, 234, 270
- Server List 25
- server list 234
- SERVER\_ENCRYPT 130, 138, 299
- SERVER\_ENCRYPT value 133
- Service Class
  - DDFBAT 111
  - DDFDEF 110
  - DDFONL 110
  - STCHI 109
- service name 73, 137
- SERVICES data set 74, 76
- SESSION 354, 385
- setDB2ClientAccountingInformation 144
- setDB2ClientApplicationInformation 144
- setDB2ClientUser 144
- setDB2ClientWorkstation 144
- SETR CLASSACT 157
- SETROPTS RACLIST 136, 154
- setup TCP/IP 70
- shared memory 85
- SHAREPORT 77, 81
- single DVIPA 234, 243
- SMF
  - Type 101 288
  - Type 30 282, 287–288
  - Type 72 282, 288
- SMF type 101 288
- SMF type 30 288
- SMF type 72 288
- SNA 12, 21, 89, 129, 280
- SPM 108, 259
- SPUFI 124, 174
- SQL 4, 35, 74, 77, 79, 81, 129, 135, 190, 236, 241, 272, 332
- SQL file 180
- SQL port 20, 55, 74, 77, 81, 86
- SQL processor 174, 199
- SQL request 5
- SQL statement 4, 94, 132
  - parameter markers 178
- SQLCA 134, 145, 333
- SQLCARD 336
- SQLCODE 181, 261
- SQLERROR 124, 191, 310
- SQLINTRP 91, 97, 224
- SQLJ 4, 34, 104, 147, 168, 249, 325
- SQLJ property 147, 167
- SQLRULES 199
- SRB 20, 105, 235, 280, 368
- SRM 105, 283
- SSID 196
- ssidDIST 35, 70–71, 156
- SSL 87, 95, 151
- SSL (secure socket layer) 87, 95, 131, 151
- SSL access 153

- SSL connection 133, 149, 151
- static SQL
  - matching 180
  - model 173, 175–176
  - package 175, 178
  - security 141, 173, 176
  - security benefit 175–176
  - security model 173
- static SQL profiling 178
- static VIPA 243
- STATICCAPFILE 178–180
- STATICLOGFILE 178
- STATICMODE 178, 180
- statistics 273, 304, 339
- statistics trace 363
- string URL 167
- Subsystem Type 107–108
- SUMMARY 288, 345, 354
- superuser 70
- SYSADM 154, 197
- SYSADM authority 199
- SYSIBM.IPNAMES 75, 86–87
- SYSIBM.LOCATION 136
  - LINKNAME columns 136
- SYSIBM.LOCATIONS 76, 86, 172
- SYSIBM.USERNAMES 87–88, 135, 139
- SYSPACKAGE 196
- Sysplex Distributor 244
- SYSPLAN 196
- SYSPLEX 235, 244
- Sysplex 25, 75–76, 238
- Sysplex Distributor 25, 55, 76–79, 243
  - connection 62, 77, 119, 243–244
  - execution results 264
  - function 99, 243, 245
  - results 261
  - spread 62, 245
  - test results 261
  - testing failover scenarios 259
  - workload balancing algorithms 62
- Sysplex workload balancing 15, 25, 234–235, 242
  - other DRDA AR client 242
  - port number 249
- SYSPRINT DD SYSOUT 159
- SYSSTAT 373
- SYSTCPDA 350
- SYSTCPIP(CTRACE) 350
- System Display and Search Facility (SDSF) 70
- System z
  - cryptographic hardware 151
  - machine 58
  - Parallel Sysplex 239
  - platform 58
- System.out.println 167
- system-directed access 186
- Systems Network Architecture 21

## T

- TALLY 354
- TCP D9C1DIST 77, 79, 81

TCP D9C2DIST 77, 79, 81  
 TCP.SC63.TCPP Arm 74  
 TCP/IP 12, 20, 35, 69–70, 129, 234, 280–281  
     data sharing 25, 61, 72, 75–76, 241, 243, 281  
     DRDA security setup 130  
     DVIPA 76, 234, 243  
 TCP/IP address  
     9.12.6.70 241  
     space 74, 243  
 TCP/IP host 71, 73, 75, 244  
 TCP/IP KeepAlive  
     configuration value 96  
     value 96  
 TCP/IP network 71, 73, 81, 132, 249, 256  
     DB2 subsystems 73  
 TCP/IP packet trace 350  
 TCP/IP port 72, 76, 86  
 TCP/IP service 20, 70, 76, 86  
 TCP/IP setup 70–71, 153  
 TCP/IP vs SNA 22  
 TCPALVER 90, 96, 131–132, 307  
 TCPCONFIG 157  
 TCPIP 74, 82, 143, 153, 169, 255  
 TCPKPALV 90, 96, 307  
 team 323  
 TEXT 179, 396  
 the environment 294–295, 332  
 The Open Group 10, 16, 462  
 thread 19, 286, 332  
 thread pooling 22  
 three-part name 187  
 ticket granting server (TGS) 137  
 TIMESTAMP 372  
 tools 29, 142, 177, 284, 287, 323  
 TRACE 195, 272, 285–286, 364  
 TRACE CT 351  
 TRACE\_ALL 338–339  
 TRACE\_CONNECTION\_CALLS 339  
 TRACE\_CONNECTS 339–340  
 TRACE\_DIAGOSTICS 339  
 TRACE\_DRDA\_FLOWS 339  
 TRACE\_DRIVER\_CONFIGURATION 339  
 TRACE\_NONE 338  
 TRACE\_PARAMETERS\_META\_DATA 339  
 TRACE\_RESULT\_SET\_CALLS 339  
 TRACE\_RESULT\_SET\_META\_DATA 339  
 TRACE\_STATEMENT\_CALLS 339  
 transaction pooling 25  
 transparency 8  
 troubleshooting 297, 341  
 TSO 91, 107, 135, 197  
 TSO batch 199  
 two-phase commit 5  
 two-phase commit protocol 5  
 Type 2 inactive thread 22

## U

UA45350 147, 397  
 UK06864 396  
 UK09411 396

UK12978 396  
 UK15499 396  
 UK15518 396  
 UK21949 396  
 UK26033 396  
 UK26257 397  
 UK32198 397  
 UK33449 396  
 UK33795 397  
 UK33843 397  
 UK35518 396  
 UK36263 397  
 UK37241 397  
 UK37310 397  
 UK38142 397  
 UK39483 398  
 UK40814 396  
 UK42748 398  
 UK42800 398  
 UK43747 398  
 UK44050 397  
 UK44299 399  
 UK44898 398  
 UK44899 398  
 UK45418 398  
 UK45981 399  
 UK46487 398  
 UK46551 398  
 UK46727 399  
 UK46942 398  
 UK47981 377, 399  
 UK48537 399  
 Unicode 125  
 UNIX System Services 20, 69–70, 85  
 UOW (unit of work) 4–5, 262  
 UPDATE 98–99, 259, 307, 329  
 update 6–7, 74, 76, 81, 135, 140, 258, 270, 284, 297, 329  
 user ID 70, 96–97, 130–131, 197, 277, 362  
     administration 140  
     DB2USER1 131  
 USER STC 70, 157–158  
 user STC  
     Digital ring information 158  
 USING 236, 302, 312, 330

## V

V TCPIP 351  
 V9.5 FP3 42, 147, 239, 242  
 Virtual Private Network (VPN) 150  
 VSAM 30, 373  
 VTAM (Virtual Telecommunication Access Method) 21,  
 85, 286, 384  
     ACB 21  
     LU 93

## W

WebSphere 42, 107, 171–172, 389  
 WebSphere Application  
     Server 63, 138, 171

WebSphere Application Server 138  
Windows 4, 33, 143, 162, 166, 259, 303–304, 329  
Wireshark 355  
WITH HOLD 23, 261, 263  
WLB 234  
WLM 56  
WLM (Workload Manager) 17, 26, 69, 105, 139, 234, 280, 344  
    classification rules 387  
    information 241  
workload balance 46, 55, 77, 233–234  
workstation name 97, 142–143  
write 6, 46, 313, 327  
wtsc63 81, 84, 143, 148, 255, 272, 300, 338

## X

XA support 42  
XA transaction 63  
    z/OS data 63  
XML 10, 26, 97, 172, 280, 312  
XQuery 44

## Z

z/OS 69–70, 130  
z/OS data 76, 120, 243  
    access 52  
    catalogued DB2 243  
    multiple workstations access DB2 53  
    Runtime Client accesses DB2 44  
z/OS server 22, 97, 104, 247  
    DB2 9 152  
z/OS subsystem 6, 104, 243  
    entire DB2 6  
    remote DB2 19  
    same DB2 243  
    single DB2 61  
z/OS sysplex  
    distributor 243  
    member 243  
z/OS V8 61, 97, 147, 265  
zIIP processor 235  
ZIIPAWMT 283

Archived



## DB2 9 for z/OS: Distributed Functions

(0.5" spine)  
0.475" <-> 0.873"  
250 <-> 459 pages









# DB2 9 for z/OS: Distributed Functions



**Redbooks®**

**Establish connectivity  
to and from DB2  
systems**

**Balance transaction  
workload across data  
sharing members**

**Explore the functions  
of Data Server Drivers  
and Clients**

Distributed Relational Database Architecture (DRDA) is a set of protocols that permits multiple local and remote database systems and application programs, to work together. Any combination of relational database management products that use DRDA can be connected to form a distributed relational database management system. DRDA coordinates communication between systems by defining what can be exchanged and how it must be exchanged.

DB2® for z/OS Distributed Data Facility (DDF) is a built-in component which provides the connectivity to and from other servers or clients over the network. DDF is a full-function DRDA compliant transaction monitor which, equipped with thread pooling and connection management, can support very large networks. Different z/OS workload management priorities can be assigned to different, user-specified classes of DDF-routed application work.

In this IBM Redbooks publication we describe how to set up your DDF environment, and how to deploy the DDF capabilities in different configurations, including how to develop applications that access distributed databases.

We also describe a set of more advanced features, such as thread pooling and high availability distributed configurations, in a DB2 data sharing environment, as well as the traces available to you to do performance monitoring and problem determination.

In summary, we show how a high-volume, highly available transactional application can be successfully implemented with a DB2 for z/OS data server accessed by all types of application servers or clients running on the same or different platform.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
**[ibm.com/redbooks](http://ibm.com/redbooks)**